uc3m | Universidad Carlos III de Madrid

Grado en ingeniería electrónica industrial y automática
2016-2017

Trabajo Fin de Grado

"Interconexión de sistemas empotrados a través de interfaces web"

Rubén Antonio Martínez Domínguez

Tutor

Raúl Sánchez Reillo

Leganés, 17 de octubre de 2017

Resumen

El proyecto que se desarrollará a lo largo de este documento consiste en el diseño e implementación de un sistema basado en la arquitectura cliente-servidor que, por un lado gestione toda la información del hotel digitalizada para el proceso, y por otro controle el acceso a las distintas instancias del edificio.

Para esto se han desarrollado dos softwares para los clientes del sistema, que consisten en una aplicación de escritorio desde la que gestionar la información ya mencionada, y una Raspberry Pi con un módulo RFID y un display LCD que actuará como cerradura. Por otra parte tendremos la parte servidor del sistema, que consistirá en una base de datos y un conjunto de scripts PHP alojados en el servidor web que nos permitirán acceder a la información de la base de datos en función de las necesidades que encontremos.

El desarrollo de este sistema supondría una clara mejora para el funcionamiento del hotel, ya que la aplicación de escritorio permitirá al empleado encargado de utilizarla añadir, eliminar o modificar cualquier información que tengamos en nuestra base de datos, permitiendo así un control instantáneo y digitalizado de todos los elementos del hotel, clientes, empleados y habitaciones. Por otra parte, supone una mejora para los clientes y empleados del hotel, dado que cada uno tendrá su tarjeta de acceso para todas las habitaciones del edificio, no existirán distintos juegos de llaves compartiéndose entre empleados y clientes. Por último, el desarrollo de este proyecto adquiere gran importancia en cuanto a que supone un coste de desarrollo considerablemente bajo, siendo un modelo de sistema que se puede extrapolar a una gran cantidad de alternativas destinadas a añadir conectividad a elementos del día a día, dando la posibilidad a aquellos interesados de implementar su propio sistema sin la necesidad de disponer de un gran presupuesto.

Executive Summary

The project that is developed throughout this document consists of the design and implementation of a system based on the client-server architecture that, on one hand, manages all the digitized hotel information for the process, and on the other, controls access to the different building rooms.

For this, two software have been developed for the clients of the system, consisting of a desktop application to manage the information already mentioned, and a Raspberry Pi with an RFID module and an LCD display that will act as a lock. On the other hand we have the server part of the system, which will consist of a database and a set of PHP scripts hosted on the web server that allows access to the information of the database according to the needs that we find.

The development of this system would be a clear improvement for the running of the hotel, as the desktop application will allow the employee in charge to add, delete or modify any information that is saved in the database, thus allowing an instantaneous and digitized control of all the elements of the hotel, clients, employees and rooms. Moreover, it is an improvement for both, clients and employees of the hotel, as each one has an access card for all the rooms of the building, there aren't different sets of keys shared between employees and clients. Finally, the development of this project acquires a great importance because it supposes a considerably low development cost, as it is a model of system that can be extrapolated to huge amount of alternatives destined to add connectivity to the elements of our everyday life, giving the possibility for those interested in implementing their own system without the need of a large budget.

Índice

Índice de Figuras

Índice de tablas

Listado de Acrónimos

API	Application Programming Interface (Interfaz de programación de aplicaciones)
ASP	Active Server Pages (Páginas de servidor activo)
BD	Base de Datos
C#	C sharp
CLI	Command-Line Interface (interfaz de línea de comandos)
DB	Data Base (Base de datos)
DNI	Documento Nacional de Identidad
GNU	GNU's Not Unix (GNU no es Unix)
GPIO	General Purpose Input/Output (Entrada/Salida de Propósito General)
GPL	General Public License (Licencia pública general)
GUTI	Grupo Universitario de Tecnologías de Identificación
Hat-Board	Hardware Attached on Top (Hardware Adjunto en la parte Superior)
HTML	HyperText Markup Language (lenguaje de marcas de hipertexto)
ID	Identificador
IDE	Integrated Development Environment (Entorno de desarrollo integrado)
IP	Internet Protocol (Protocolo de internet)
LCD	Liquid Cristal Display (Representación visual por cristal líquido)
MacOS	Macintosh Operating System (Sistema operativo de Macintosh)
NET	Network (Red)
NFC	Near Field Communication (Comunicación de campo cercano)
NOOBS	New Out Of Box Software

PHP Hipertext Preprocessor (Preprocesador de hipertexto)

PIN Personal Identification Number (Número de identificación personal)

RFID Radio Frequency Identification (Identificación por radiofrecuencia)

SPI Serial Peripheral Interface (Interfaz de periféricos serie)

SQL Structured Query Languaje (lenguaje de consulta estructurada)

TCP/IP Trasmission Control Protocol / Internet Protocol

URL Uniform Resource Locator (Identificador de recursos uniforme)

UTF Unicode Transformation Format (Formato de codificación de caracteres Unicode)

XML eXtensible Markup Language (Lenguaje de marcas extensible)

1 Introducción

1.1 Motivación y objetivos

Este proyecto surge, por una parte, por el interés en una investigación sobre la tecnología relacionada con Raspberry Pi y su conexión tanto con otros dispositivos como con diversos componentes que se pueden agregar a su hardware para ofrecer un gran abanico de posibilidades, y por otra, por el interés en estudio de las tecnologías orientadas a actuar como interfaz web. Estos intereses nacen tras haber manejado escasos sistemas electrónicos reales después de haber finalizar un grado en ingeniería electrónica industrial y automática, debido a esto se pretende tanto encontrar la forma de aplicar conceptos teóricos adquiridos durante el grado a una situación real como aprender sobre otras tecnologías no estudiadas durante la carrera.

El objetivo principal de este proyecto es diseñar e implementar un sistema que permita llevar un control sobre toda la información de clientes, empleados y habitaciones de un hotel y utilizar dicha información para poder controlar el acceso a las distintas instancias del hotel correctamente.

Por otra parte, se establecen una serie de objetivos secundarios que aporten funcionalidad supongan una mejora sobre el sistema:

- ✓ Utilizar un servidor web para alojar toda la información almacenada
- ✓ Crear una aplicación de gestión de la información con una interfaz intuitiva
- ✓ Agregar un método de comunicación del usuario con el sistema de control de acceso

1.2 Entorno socio-económico y marco regulador

En lo relacionado con el marco regulador que engloba el desarrollo de este proyecto, es un punto importante considerar la normativa de electrónica de consumo, controlando los aspectos fundamentales de seguridad o corriente utilizada necesarios cuando se trabaja con un equipo electrónico. Por otra parte, todos los datos utilizados se encontrarán protegidos bajo la ley de protección de datos para asegurar el correcto uso de los mismos y proporcionar así una mayor seguridad a los clientes. Por último, allá donde sea necesario se aplicarán los estándares

pertinentes, como el protocolo TCP/IP, que se aplicará en la parte de conexión a internet desde los clientes del producto para mejorar su conectividad y compatibilidad con otros equipos o ante cambios realizados en el sistema.

En cuanto el entorno socio-económico, se ha de destacar el bajo precio de los componentes utilizados en el proyecto, especialmente considerando sus grandes capacidades y su elevada potencia que permite generar una gran variedad de sistemas tanto a particulares como a pequeñas empresas, permitiendo así ampliar las oportunidades de negocio de muchos entornos. También se ha de tener en cuenta que este proyecto consiste en añadir conectividad a aparatos o sistemas que un usuario normal puede utilizar en sus día a día, lo que se conoce como "internet of things" o el internet de las cosas. Esto, y el hecho de que la creación de un programa que acceda a cierta información alojada en una base de datos y realice una acción con dicha información, se puede aplicar a una gran variedad de problemas con distintas funciones, influye también en el aumento de la calidad de vida del ciudadano medio al tener cada vez más a su alcance un proceso por el cual se automatizan o digitalizan acciones del día a día haciéndolo más cómodo y fácil.

1.3 Estructura del documento

Este documento se encuentra dividido en distintos capítulos que aportan la información necesaria para una correcta comprensión del proyecto, así como de sus distintas tecnologías y funcionamiento. Podemos ver el ciclo de vida del proyecto en la figura 1.1, y a continuación se presenta una breve descripción del contenido de cada capítulo.

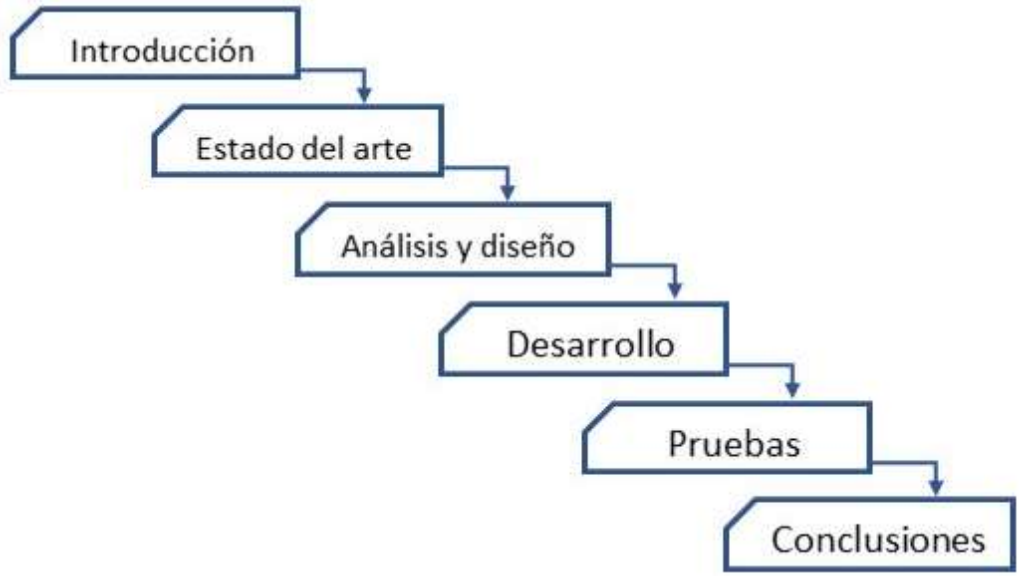


Figura 1.1: Ciclo de vida del proyecto

- Capítulo 1 - Introducción: En este capítulo se presenta una pequeña exposición de los objetivos y motivaciones que han propiciado la creación del mismo, así como una presentación del entorno socio-económico y el marco regulador que influye en el desarrollo del proyecto y un apartado que resume la estructura seguida por el documento.

- Capítulo 2 - Plataformas de desarrollo de servidor: A lo largo de este capítulo se ofrece una visión global de aquellas plataformas que se han utilizado posteriormente para desarrollar todo lo relacionado con la parte servidor del sistema, es decir, la base de datos, el servidor utilizado y los scripts PHP para comunicarse. Este capítulo junto con el capítulo 3 conforman un análisis del estado del arte del proyecto.
- Capítulo 3 - Plataformas de desarrollo de cliente: Durante este capítulo se muestra toda la información necesaria para comprender las plataformas de desarrollo empleadas en la parte de los distintos clientes del sistema, la aplicación de escritorio destinada a gestionar toda la información del hotel y la Raspberry Pi que actuará a modo de cerradura.
- Capítulo 4 - Análisis y diseño del sistema: Este capítulo recoge tanto el estudio de requisitos y posibles problemas a abordar en el desarrollo del sistema como la búsqueda de posibles soluciones a estos, analizándolos tanto desde un punto de vista global, buscando la mejor forma de proceder entre las disponibles como el método a seguir dentro de la opción elegida.
- Capítulo 5 - Desarrollo del sistema: Aquí se mostrará toda la información necesaria para una correcta comprensión de la fase de desarrollo del sistema, enfatizando en aquellas partes más importantes o específicas y mostrando una parte del código utilizado en algunos casos en los que fuese relevante.
- Capítulo 6 - Pruebas de funcionamiento del sistema: En este capítulo se presentan todas aquellas pruebas realizadas sobre la aplicación de escritorio y sobre el programa de la Raspberry, así como el resultado obtenido ante dichas pruebas.
- Capítulo 7 - conclusiones: Por último, en este capítulo se expondrán las conclusiones obtenidas de la realización del proyecto, analizando los objetivos planteados y alcanzados y estudiando los posibles avances del sistema.

2 Plataformas de desarrollo de servidor

En este capítulo se describirán las plataformas utilizadas en el desarrollo de la parte de la base de datos, el servidor utilizado para alojarla y los scripts empleados para conectar con ella, presentando una breve definición de cada una y algunos ejemplos de ejecución simples orientados a facilitar la comprensión de las mismas.

2.1 Bases de datos

Una base de datos [1] se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos, que permiten el acceso directo a ellos y un conjunto de programas que manipulen dichos datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, mientras que cada fila conforma un registro de la tabla.

2.1.1 Sistema de gestión de datos SQL

Los Sistemas de Gestión de Base de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, un lenguaje de manipulación de los mismos y un lenguaje de consulta.

En este proyecto hemos utilizado el sistema de gestión de datos SQL, debido a que mantiene una mayor concordancia con la tecnología de Microsoft utilizada para el proyecto. El SQL [2] es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales y permite especificar diversos tipos de operaciones en ellos. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

En este proyecto hemos empleado el lenguaje SQL referido principalmente a las siguientes sentencias pertenecientes al Lenguaje de manipulación de datos:

1. La sentencia **SELECT** nos permite consultar los datos almacenados en una tabla de la base de datos. Tiene la siguiente estructura básica, cuyos atributos en cursiva no son obligatorios.

   SELECT [ALL | DISTINCT] <nombre_campo> [{,<nombre_campo>}]

   FROM <nombre_tabla>|<nombre_vista>

   [WHERE <condición> [{ AND|OR <condición>}]]

   [GROUP BY <nombre_campo> [{,<nombre_campo >}]]

   [HAVING <condición>[{ AND|OR <condición>}]]

   [ORDER BY <nombre_campo>|<indice_campo> [ASC | DESC]

   SELECT: Palabra clave que indica que la sentencia de SQL que queremos ejecutar es deselección.

   ALL: Indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca. También puede utilizarse * para seleccionar todos los valores.

   DISTINCT: Indica que queremos seleccionar sólo los valores distintos.

   FROM: Indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o "join". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

   WHERE: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR.

   GROUP BY: Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

   HAVING: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

   ORDER BY: Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

   EJEMPLO 1:

   Para formular una consulta a la tabla Coches y recuperar los campos matricula, marca, modelo, color, numero_kilometros, num_plazas debemos ejecutar la siguiente consulta. Los datos serán devueltos ordenados por marca y por modelo en orden ascendente, de menor a mayor. La palabra clave FROM indica que los datos serán recuperados de la tabla Coches.

```
SELECT matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM Coches
ORDER BY marca, modelo;
```

EJEMPLO 2:

Para formular una consulta a tablas relacionadas, como es el caso de las tablas prueba_estudios_edad y prueba_2 que tienen una relación entre el campo ***Estudios*** de la tabla prueba_2 y el campo ***ID*** de la tabla prueba_estudios_edad debemos ejecutar la consulta con la siguiente estructura. Los datos devueltos serán todos los nombres de la tabla prueba_2 y todas las edades de la tabla prueba_estudios_edad.

```
SELECT prueba_2.Nombre , prueba_estudios_edad.Edad
FROM prueba_estudios_edad JOIN prueba_2
ON prueba_2.Estudios = prueba_estudios_edad.ID;
```

2. La sentencia **INSERT** de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional. Tiene la siguiente estructura básica:

```
INSERT INTO 'nombre_tabla' ('columna1',['columna2,... '])
VALUES ('valor1', ['valor2,...'']);
```

O también se puede utilizar como (solo cuando en VALUES introduces TODAS las columnas de la tabla):

```
INSERT INTO 'nombre_tabla' VALUES ('valor1','valor2');
```

EJEMPLO 1:

Para introducir una nueva fila en la tabla agenda_telefonica con los valores “Roberto Jeldrez” en el campo nombre y “4886850” en el campo numero debemos ejecutar la siguiente consulta:

```
INSERT INTO agenda_telefonica (nombre, numero)
VALUES ('Roberto Jeldrez', 4886850);
```

EJEMPLO 2 (asumiendo que 'nombre' y 'número' son las únicas columnas de la tabla 'agenda_telefonica')**:**

```
INSERT INTO agenda_telefonica
VALUES ('Roberto Jeldrez', 4886850);
```

3. La sentencia **UPDATE** de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla. Tiene la siguiente estructura básica:

```
UPDATE nombre_tabla
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

EJEMPLO:

Para modificar los campos Nombre y DNI de la tabla clientes en aquellas filas donde la columna Apellidos coincida con Martínez, debemos ejecutar la siguiente consulta:

```
UPDATE clientes
SET Nombre = ’Nuevo_nombre’, DNI = ’Nuevo_DNI’
WHERE Apellidos = ‘Martinez’;
```

4. La sentencia **DELETE** de SQL borra uno o más registros existentes en una tabla. Tiene la siguiente estructura básica:

```
DELETE FROM nombre_tabla WHERE columna1 = 'valor1';
```

EJEMPLO:

Para borrar uno o varios registros de la tabla aulas en los que numero_pizarras sea 2 ejecutaremos la siguiente sentencia:

```
DELETE FROM aulas WHERE numero_pizarras = ‘2’;
```

2.1.2 Servidor XAMPP

XAMPP [3] es un servidor independiente multiplataforma, de software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.

El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOS X.

XAMPP se actualiza regularmente para incorporar las últimas versiones de Apache/MySQL/PHP y Perl. También incluye otros módulos como Open SSL y phpMyAdmin.

El servidor XAMPP fue diseñado para facilitar el testeo de sitios web y diversos programas, sin embardo es utilizado como servidor de sitios web debido a que aporta una seguridad suficiente y algunas herramientas para proteger las partes más importantes.

Instalación de XAMPP

A continuación, mostraremos el proceso de instalación de XAMPP:

Al poner en marcha el instalador XAMPP nos muestra dos avisos:

El primero aparece si en el ordenador hay instalado un antivirus como muestra la figura 2.1:

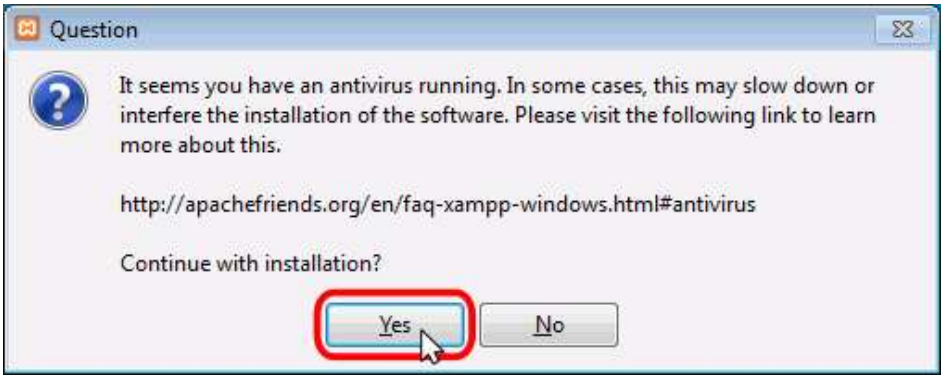


Figura 2.1: Aviso de antivirus [15]

El segundo aparece si está activado el Control de Cuentas de Usuario y recuerda que algunos directorios tienen permisos restringidos, podemos verlo en la figura 2.2:

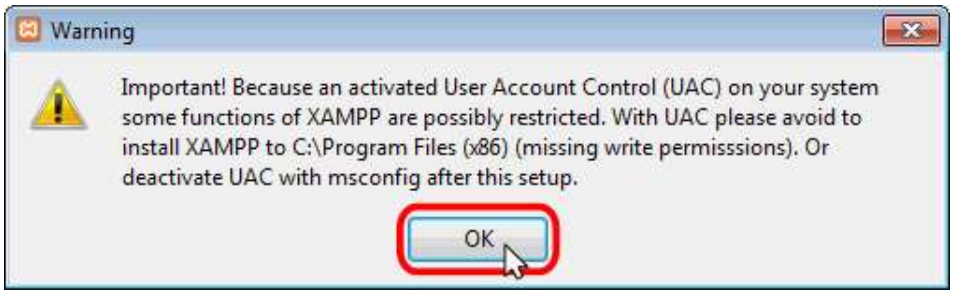


Figura 2.2: Aviso de permisos restringidos [15]

A continuación se inicia el instalador de XAMPP, accederemos a la siguiente ventana, en la que se seleccionarán los componentes que se van a instalar pulsando el botón “Next”.

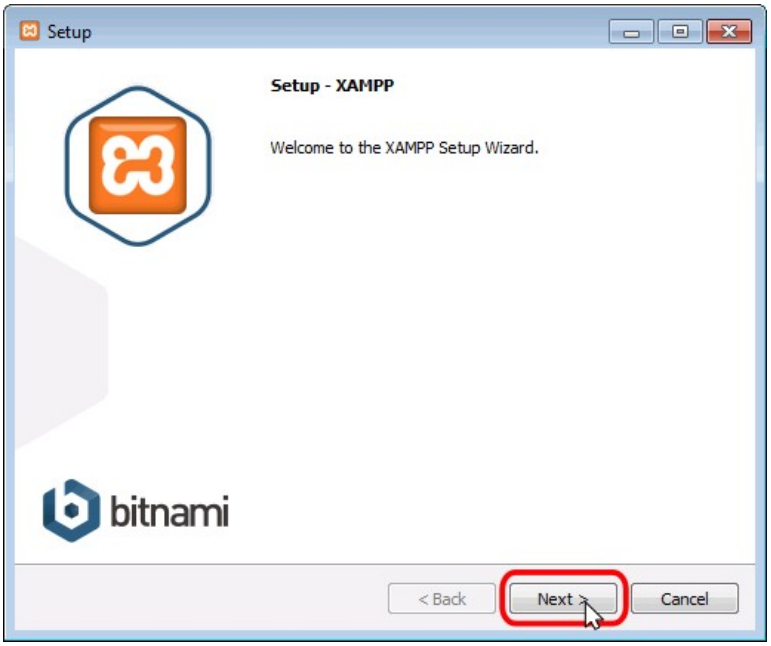


Figura 2.3: Inicio instalador [15]

XAMPP instalará el servidor Apache y el lenguaje PHP como contenido mínimo, sin embargo, podemos elegir instalar otros componentes seleccionando el cuadro a la izquierda de los mismos que aparecen en una lista como se muestra en la figura 2.4. En la instalación de XAMPP para el desarrollo de este proyecto, se instalaron todos los componentes.

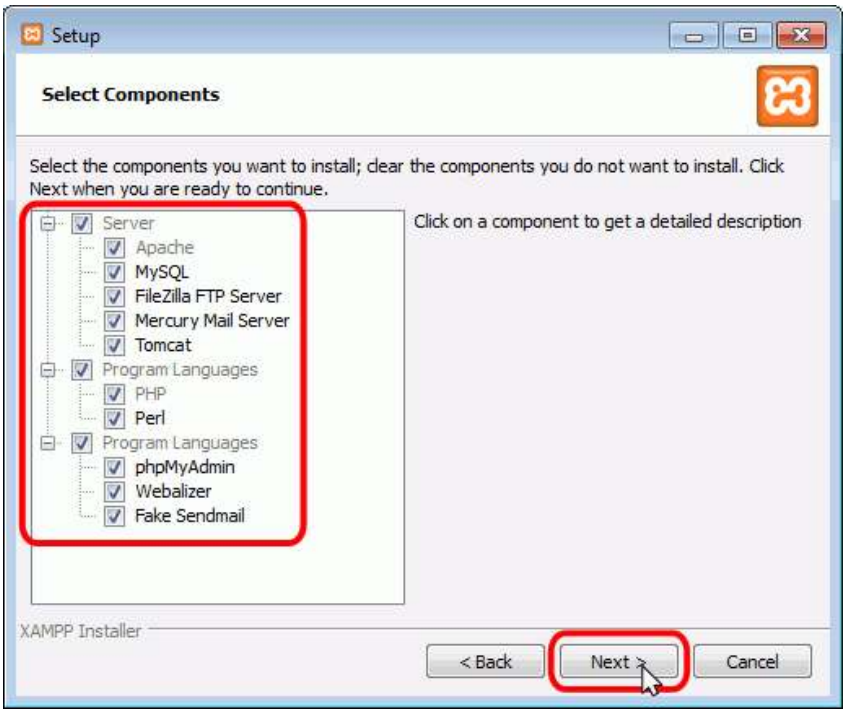


Figura 2.4: Selección de componentes en instalador [15]

En la siguiente pantalla seleccionaremos el directorio de la carpeta de instalación de XAMPP, el directorio por defecto y seleccionado en la instalación de XAMPP es C:\xampp. Podemos verlo en la figura 2.5:

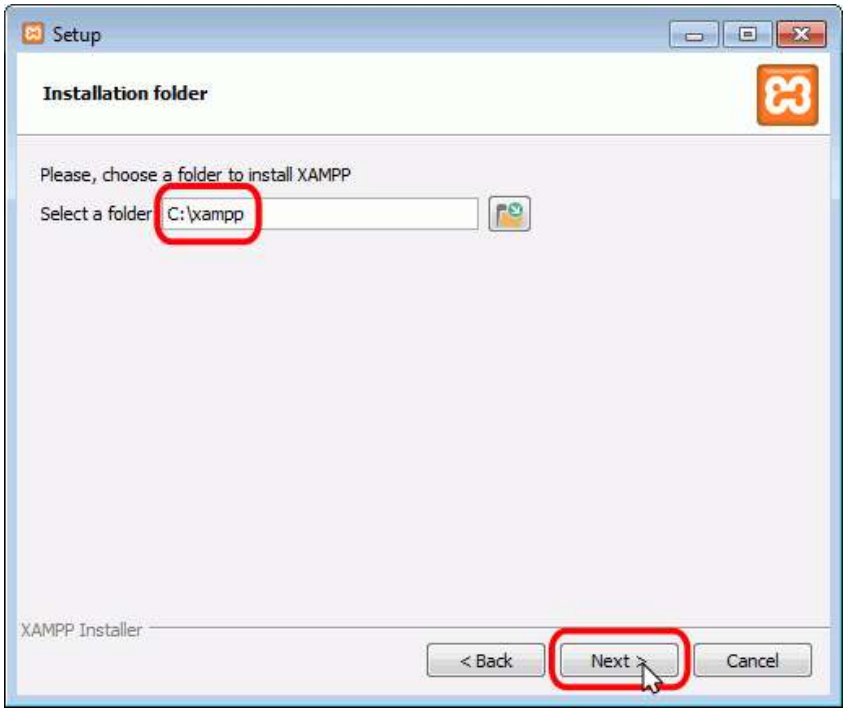


Figura 2.5: Selección de directorio en instalador [15]

La siguiente ventana del instalador nos muestra un mensaje informando de la existencia de los instaladores de aplicaciones para XAMPP creados por Bitnami y un cuadro desmarcable que nos abrirá una página con documentación sobre estas aplicaciones. Para continuar pulsar el botón “Next”.

La configuración del instalador ya está lista, pulsaremos el botón “Next” de la siguiente ventana para comenzar la instalación, la cual observamos en la figura 2.6:

Figura 2.6: Instalación en curso [15]

Durante la instalación, si en el ordenador no se había instalado Apache anteriormente, se mostrará un aviso del cortafuegos de Windows para autorizar a Apache para comunicarse en las redes domésticas o de trabajo, lo que debemos permitir haciendo clic en el botón "Permitir acceso" como se muestra en la figura 2.7.

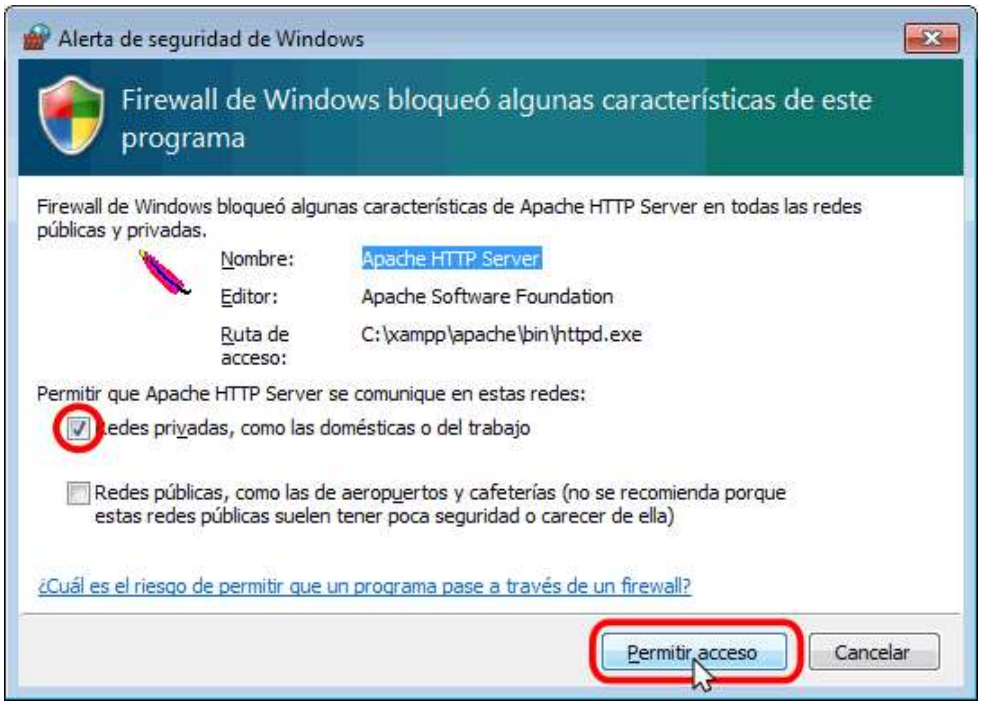


Figura 2.7: Aviso cortafuegos [15]

Una vez termine la instalación de XAMPP, el instalador nos mostrará la última ventana informándonos de que la instalación ha sido completada y mostrando un mensaje con su correspondiente cuadro desmarcable para abrir el panel de control de XAMPP al hacer click en "Finish" y cerrar el instalador. Podemos ver esta última ventana en la figura 2.8.

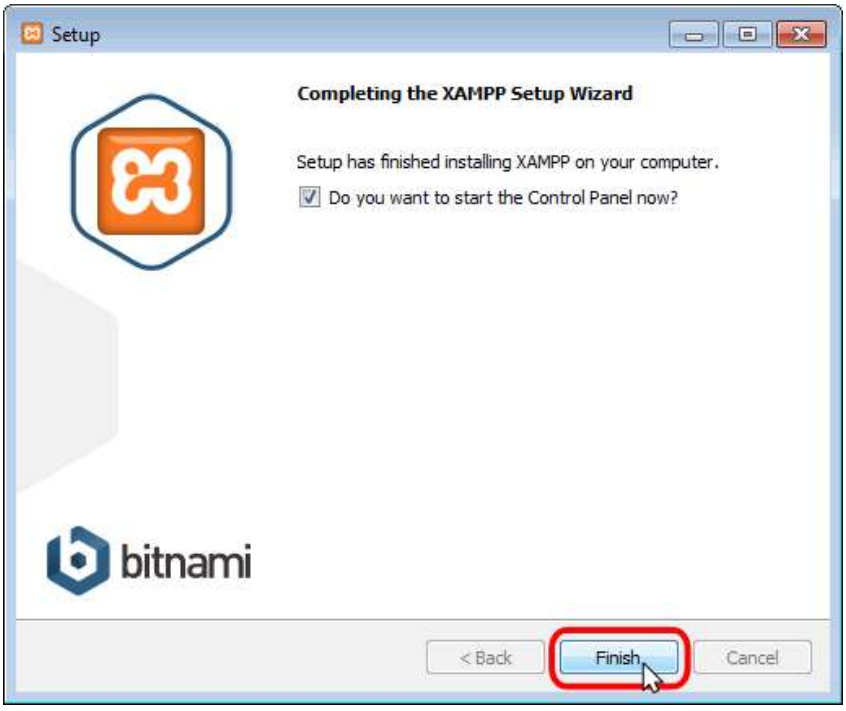


Figura 2.8: Final de instalación [15]

2.2 PHP

El lenguaje PHP [4] es un lenguaje de código abierto normalmente dedicado al desarrollo web y que puede incluso ser incrustado en lenguaje HTML. Utilizado también en servidores para gestionar llamadas HTTP, lo que es perfecto para alcanzar los objetivos de este proyecto.

En este trabajo se ha utilizado el lenguaje PHP para procesar la petición de un usuario mediante la interpretación de un script en el servidor web para obtener respuestas dinámicamente. Más concretamente se ha utilizado como enlace entre la aplicación desarrollada en C# y los datos de la base de datos disponibles en MySQL.

A continuación, se mostrarán algunos ejemplos de archivos PHP explicados:

Primero debemos ejecutar el servidor XAMPP y lanzar el servicio MySQL. Con el servidor corriendo debemos tener el archivo a ejecutar en la carpeta htdocs de XAMPP y ejecutarlos desde el navegador escribiendo la siguiente ruta URL: localhost/nombre_archivo.php

EJEMPLO 1:

Ahora vamos a ver un ejemplo simple de ejecución de un archivo PHP para mostrar “hola mundo” en nuestro navegador.

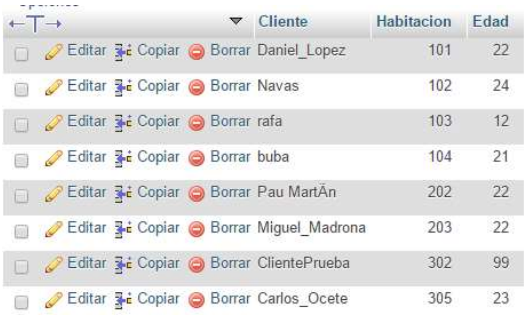
```
<?php
echo 'Hola mundo.';
?>
```

Podemos ver el código del script y ejecución del mismo en la figura 2.9:

Figura 2.9: Código y ejecución de ejemplo 1 PHP

EJEMPLO 2:

en el siguiente ejemplo vamos a ver un script que nos devuelve todos los campos de la tabla clientes. En la figura 2.10 tenemos los datos de la tabla clientes para comparar los resultados del código de ejecución, que se puede ver en la figura 2.11 con la ejecución del mismo en la figura 2.12:



	Cliente	Habitacion	Edad
Editar Copiar Borrar	Daniel_Lopez	101	22
Editar Copiar Borrar	Navas	102	24
Editar Copiar Borrar	rafa	103	12
Editar Copiar Borrar	buba	104	21
Editar Copiar Borrar	Pau MartÃn	202	22
Editar Copiar Borrar	Miguel_Madrona	203	22
Editar Copiar Borrar	ClientePrueba	302	99
Editar Copiar Borrar	Carlos_Ocete	305	23

Figura 2.10: Contenido tabla clientes en ejemplo 2 PHP

prueba3.php: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?php
$dbhost = "localhost";
$dbuser = "root";
$dbpass = "";
$dbHabitacion = "amoaproba";

//Connecting to DB
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die('Error connecting to mysql');
mysql_select_db($dbHabitacion);

$query  = "SELECT * FROM clientes";
$result = mysql_query($query);
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
echo "<MessageXML>";
while($row = mysql_fetch_array($result, MYSQL_ASSOC))
{
  echo "cliente: ";
  echo "<Data>"."<Cliente>{$row['Cliente']}</Cliente> "."</Data>";
  echo "Habitacion: ";
  echo "<Data>"."<Habitacion>{$row['Habitacion']}</Habitacion>"."</Data>";
  echo "        Edad: ";
  echo "<Data>"."<Edad>{$row['Edad']}</Edad>   "."</Data>"."</Data>";
  echo "<br></br>";
}
echo "</MessageXML>";
mysql_close($conn);
?>
```

servidor
usuario
contraseña
Base de datos

Declaramos las variables que vamos a utilizar posteriormente para llamar a la base de datos sin introducir directamente los datos de la misma.

Sentencia a consultar a la base de datos

Impresión de la información obtenida de los distintos campos de cada fila de la tabla clientes.

Figura 2.11: Código de ejemplo 2 PHP

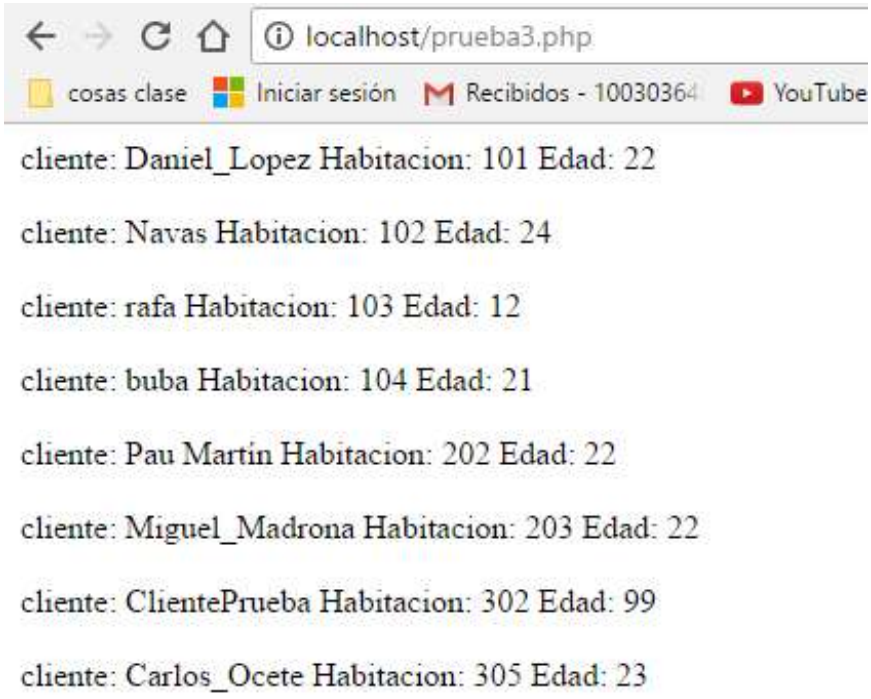


Figura 2.12: Ejecución de ejemplo 2 PHP

EJEMPLO 3:

En este ejemplo vamos a ver un script que recibe Cliente, Habitación y Edad como datos a introducir en la tabla clientes desde una aplicación en C#.

En la figura 2.13 podemos ver el código PHP:

insert.php: Bloc de notas

Archivo Edición Formato Ver Ayuda

```php
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "amoaproba");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Escape user inputs for security
$Cliente = mysqli_real_escape_string($link, $_POST['Cliente']);
$Habitacion = mysqli_real_escape_string($link, $_POST['Habitacion']);
$Edad = mysqli_real_escape_string($link, $_POST['Edad']);

// attempt insert query execution
$sql = "INSERT INTO clientes (Cliente, Habitacion, Edad) VALUES ('$Cliente', '$Habitacion', '$Edad')";
if(mysqli_query($link, $sql)){
    echo "Records added successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}

// close connection
mysqli_close($link);
?>
```

Conexión con la base de datos

Asignación de las variables que recibe el archivo a variables tipo PHP para utilizarlas posteriormente

Declaración de la consulta

Llamada a la base de datos con la consulta generada

Figura 2.13: Código de ejemplo 3 PHP

Podemos observar el estado de la base de datos antes y después de pulsar el botón “Nuevo cliente” en las figuras 2.14 y 2.15 respectivamente:

Figura 2.14: Tabla clientes antes de ejecución de ejemplo 3 PHP

Figura 2.15: Tabla clientes después de ejecución de ejemplo 3 PHP

3 Plataformas de desarrollo de clientes

En este capítulo se describirán las plataformas utilizadas en el desarrollo de la parte de los clientes del sistema, presentando una breve definición de cada una y algunos ejemplos de ejecución simples orientados a facilitar la comprensión de las mismas.

3.1 Microsoft Visual Studio y C#

Visual Studio [5] es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML.

C# es un lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado como parte de su plataforma .NET, [6] derivando su sintaxis, al igual que Java, de lenguajes como C/C++ y utilizando el modelo de objetos similar al de JAVA, proveniente de la plataforma .NET, aunque incluye mejoras basadas en otros lenguajes. Es un lenguaje de programación independiente diseñado para generar software únicamente para la anteriormente nombrada plataforma .NET. Su gestión de memoria es automática y fácilmente portable gracias a la compilación del software a código intermedio, características similares al anteriormente analizado lenguaje JAVA. El entorno de desarrollo integrado más utilizado que soporta este tipo de lenguaje es Microsoft Visual Studio. La limitación más destacable es que se trata de un entorno de programación que únicamente está orientado a sistemas operativos Windows.

El desarrollo de la aplicación con la que interactuará el usuario del hotel encargado del puesto de recepción o control del edificio se ha desarrollado en Visual Studio mediante la herramienta de Windows Forms, una de las interfaces de programación de aplicación gráfica (API) que se incluyen como parte de Microsoft .NET Framework.

Con Windows Forms, puede desarrollar aplicaciones smart client [7]. Las aplicaciones smart client son aplicaciones gráficamente enriquecidas, fáciles de implementar y actualizar, que pueden trabajar con o sin conexión a Internet y que pueden tener acceso a los recursos del equipo local de un modo más seguro que las aplicaciones tradicionales basadas en Windows.

Desarrollar la aplicación en C# permite generar directamente una interfaz gráfica con la forma de una ventana de Windows a la que le podemos agregar distintos elementos. Estos elementos están separados en varias categorías accesibles desde la barra "cuadro de herramientas":

- Controles comunes: en esta categoría podemos encontrar aquellos elementos que más veces se insertan en una aplicación o cuya funcionalidad es básica para casi cualquier programa que queramos crear. En esta sección encontramos objetos como botones, cuadros de texto, cuadros para etiquetas y otros elementos utilizados en nuestro proyecto que se explicarán más adelante.
- Contenedores: en la categoría de contenedores podemos encontrar distintos elementos que nos permiten agrupar y ordenar otros elementos que podemos introducir dentro de nuestros contenedores o bien en el contexto de nuestra ventana principal de la aplicación.
- Menús y barras de herramientas: aquí tenemos varios elementos que muestran información sobre nuestro proyecto en distintos modelos de menús o barras con las que interactuar con los elementos de nuestra aplicación.
- Datos: en esta sección encontraremos todos aquellos elementos que nos sirvan para manejar los datos existentes en nuestro programa, así como para generar datos nuevos o incluso representar una caché de datos en memoria.
- Componentes: en los componentes podemos encontrar elementos como un cuadro para interactuar con los eventos de Windows, acceso a una cosa de mensajes o temporizadores para controlar tiempos o generar eventos a intervalos fijos controlados por el usuario. Todos estos elementos nos aportan información sobre el comportamiento de la aplicación y nos permite controlarlo más fácilmente.
- Impresión: encontraremos todas las herramientas necesarias para poder configurar o ajustar una impresión en papel de nuestro documento o alguna parte del mismo.
- Cuadros de diálogo: por último, en la categoría de cuadros de diálogo encontraremos todos aquellos cuadros de diálogo con los que el usuario debería interactuar para guardar o abrir un documento, seleccionar una carpeta para ello o definir la fuente de un párrafo.

Una vez en el entorno de programación de Visual Studio 2017 elegimos el tipo de aplicación que queremos crear, en nuestro caso una aplicación de Windows Forms en C#.

Cuando se genera e inicia el documento que acabamos de crear, lo primero que vemos es una ventana de Windows, cuyo nombre por defecto viene definido como "Form1". Esta ventana podemos modificarla gráficamente desde la pestaña de [Diseño], insertando y modificando los elementos que deseemos desde el cuadro de herramientas. Para modificar estos elementos, programarles unas rutinas o comportamientos o modificarlos desde el nivel de programación de los mismos accederemos a la pestaña Form1.cs, donde encontramos la definición de cada uno de estos elementos. Podemos ver todo esto en una aplicación vacía en la figura 3.1:

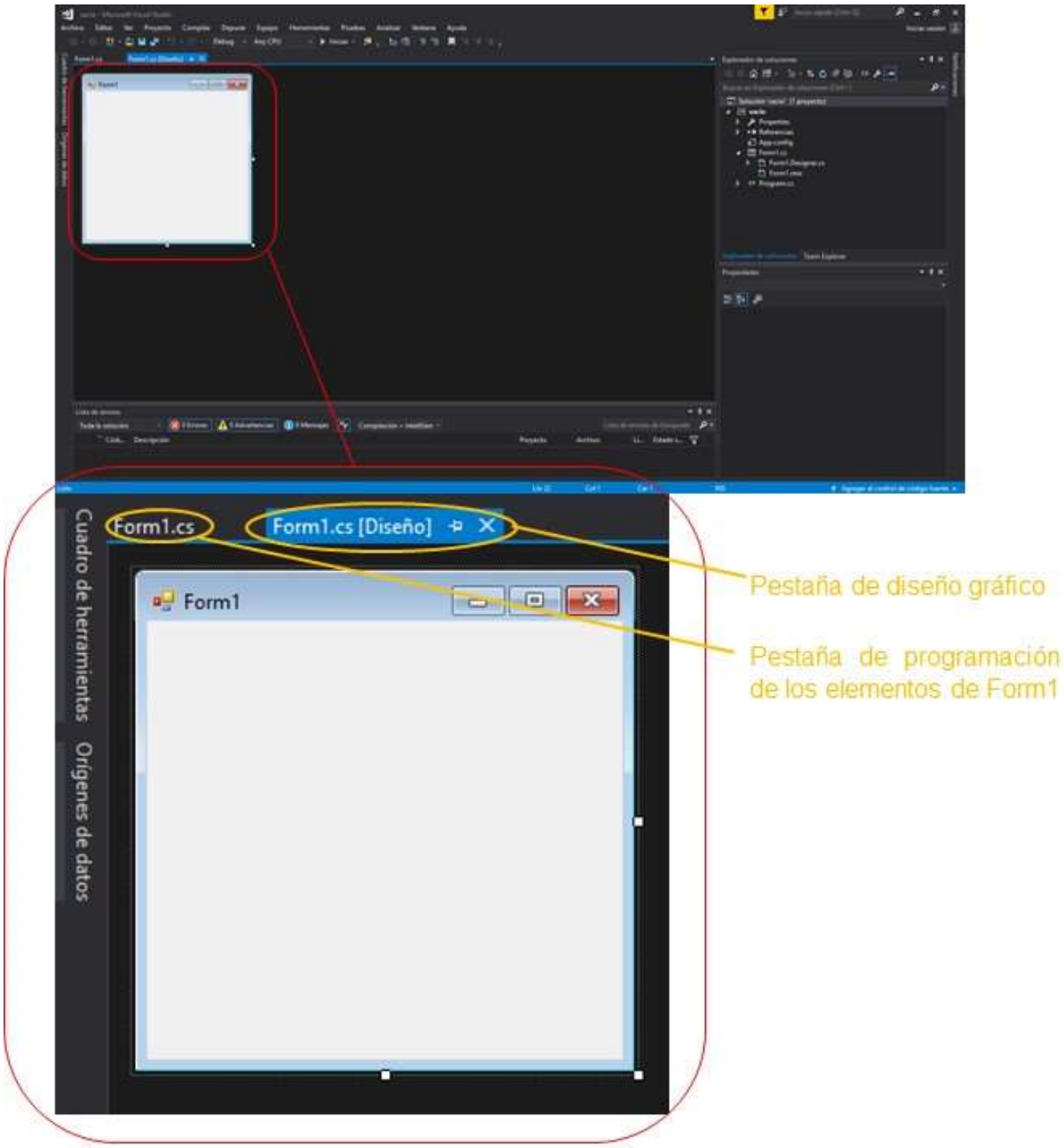


Figura 3.1: Ejemplo de aplicación de Windows Forms vacía

A continuación, veremos un ejemplo de cómo crear una aplicación sencilla en Visual Studio con Windows Forms en C#.

EJEMPLO:

La primera aplicación que vamos a crear consta únicamente de un cuadro de texto, un marco con un mensaje y dos botones como vemos en la figura 3.2:

Figura 3.2: Ejemplo 1 de aplicación de Windows Forms

El marco con mensaje es el que contiene la frase "Introducir nombre: ", la herramienta utilizada es la llamada *Label*, y únicamente nos permite introducir un texto descriptivo. Tenemos otro Label oculto debajo del primero que mostrará un mensaje de saludo cuando se pulse el botón 'Saludar'.

El cuadro con fondo blanco en el que pone "Este es un texto de prueba" es un *TextBox*, esta herramienta es de las más comúnmente utilizadas, tanto para mostrar mensajes como para introducir información en el programa. El mensaje que muestra es un texto predefinido y modificable desde las propiedades del elemento y será el texto que recoja el botón para ejecutar la función que le hemos implementado.

Por último tenemos los dos botones, 'Saludar' y 'Cerrar'. En estos botones hemos implementado dos pequeñas funciones, una para mostrar un mensaje en el Label2 que salude recogiendo el nombre que haya escrito en el TextBox. A continuación mostraremos el código de éste para ver como se implementaría una función en un botón en la figura 3.3:

```
private void B1_Click(object sender, EventArgs e)
{
    L2.Text = "Hola, " + TB1.Text;
}
```

Figura 3.3: Código de botón “Saludar” de ejemplo de C#

Para definir la función B1_Click, que es la que se inicia al hacer click en el botón ‘Saludar’ tan solo debemos hacer doble click y el programa nos llevará a la declaración del botón en la pestaña de programación de nuestro Form1, Form1.cs.

La función es muy simple, cuando se activa asigna al texto del Label2 (llamado L2) una cadena que empieza por “Hola, “ y a continuación el texto recogido del TextBox (llamado TB1).

Después tenemos el botón de cerrar, cuyo código podemos ver en la figura 3.4:

```
private void B2_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Figura 3.4: Código del botón “Cerrar” de ejemplo de C#

Esta función nos cierra el Form en el que nos encontremos actualmente, y por tanto cierra el programa. ‘this’ es un puntero que indica la selección del elemento en el que nos encontramos, por tanto al emplear this.close(); cerramos el objeto Form1.

3.2 Sistema operativo Raspbian

Raspbian [8] es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en Debian Jessie para la placa computadora Raspberry Pi, orientado a la enseñanza de informática. Contiene herramientas de desarrollo como aplicaciones para el lenguaje de programación Python o Scratch, y diferentes ejemplos de juegos usando los módulos Pygame. Destaca también el menú "raspi-config" que permite configurar el sistema operativo sin tener que modificar archivos de configuración manualmente. Entre sus funciones, permite expandir la partición root para que ocupe toda la tarjeta de memoria, configurar el teclado o aplicar overclock entre otras.

Al ser una distribución de GNU/Linux las posibilidades son infinitas. Todo software de código abierto puede ser recompilado en la propia Raspberry Pi para arquitectura armhf que pueda ser utilizado en el propio dispositivo en caso de que el desarrollador no proporcione una versión ya compilada para esta arquitectura.

3.2.1 Instalación de sistema operativo Raspbian

Existen dos métodos de instalación del sistema operativo Raspbian en la placa Raspberry Pi, utilizar el software NOOBS distribuido desde la página oficial de Raspberry e instalar el sistema operativo desde su asistente de instalación, o bien a partir de la instalación de imágenes. Dado que la instalación mediante NOOBS es la recomendada por Raspberry Pi Fundation para usuarios nuevos en este entorno, se detallará este proceso a continuación:

- Descargar el software NOOBS desde raspberrypi.org/downloads.
- Formatear una tarjeta SD de al menos 4GB en formato FAT.
- Descargar y extraer los archivos del archivo comprimido NOOBS.
- Copiar los archivos extraídos en la tarjeta SD.
- Instalación e implementación del software requerido
- En el primer arranque de NOOBS, la partición Recovery FAT será reducida al mínimo y una lista de los sistemas operativos se desplegarán.
- Elegir el sistema operativo Raspbian.
- Esperar la finalización de la instalación

Configuración inicial:

Al iniciar el sistema operativo por primera vez, el sistema mostrará el menú de configuración "raspi-config" desde el cual se podrán indicar o modificar las opciones básicas del sistema operativo.

Una vez terminada la configuración inicial en función de las necesidades del usuario el sistema operativo ya estará listo para ser utilizado.

3.3 Python

Python es un lenguaje de programación de código abierto de alto nivel, interpretado y multiplataforma cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje interpretado dado que no se debe ejecutar el código antes de su ejecución, dado que esta compilación es realizada de forma transparente al usuario por un intérprete, que lo analiza y ejecuta.

El lenguaje de programación Python fue desarrollado inicialmente para Unix, pero hoy en día hay versiones de Python disponibles en múltiples plataformas, y cualquier sistema es compatible con él mientras exista un intérprete programado para ello.

En los últimos años ha cobrado una gran popularidad, debido a varias razones como:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje.
- La sencillez y velocidad con que se crean los programas favorece el aprendizaje en usuarios principiantes y supone una alternativa más simple y rápida para usuarios expertos.
- La cantidad de plataformas en las que podemos desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.

- Python es un software de código abierto y, por tanto, gratuito. Al ser un software libre tiene una gran comunidad de usuarios que aportan continuas mejoras del lenguaje, así como una activa participación en foros de ayuda.

Por todas estas características, su simplicidad, potencia y el apoyo de una comunidad en rápido crecimiento, Python es una excelente opción para crear todo tipo de programas que se ejecuten en casi cualquier máquina.

4 Análisis y diseño del sistema

En este capítulo se presenta la fase de análisis y diseño de todo el sistema, desde las distintas partes del mismo hasta la comunicación entre ellas, así como un estudio superficial de otras opciones para desarrollar algunas de las funciones del sistema.

A lo largo del capítulo se aclararán tanto las funcionalidades necesarias de cada parte del sistema como la forma que se ha buscado para solucionar estos requerimientos.

4.1 Arquitectura del sistema

El sistema se compondrá de dos puntos físicos, uno el equipo en el que se controlará y supervisará el sistema mediante una aplicación de escritorio y otro representado por todas las cerraduras de cada una de las puertas del hotel, y una base de datos que contendrá toda la información que necesiten estos dos puntos para realizar sus correspondientes funciones. Por esto la arquitectura de la aplicación se basará en un modelo cliente servidor, el cual podemos ver en la figura 4.1.

Cada uno de esos puntos físicos constituye la parte cliente y la base de datos se alojará en un servidor, a la que los clientes accederán mediante llamadas http.

Un servidor [9] es una aplicación que ofrece un servicio a usuarios de Internet; un cliente es el que pide ese servicio. Una aplicación consta de una parte de servidor y una de cliente, que se pueden ejecutar en el mismo o en diferentes sistemas.

Para este sistema, necesitamos acceder a la base de datos desde distintos puntos, de modo que esta deberá estar alojada en un servidor web para acceder a ella desde los clientes.

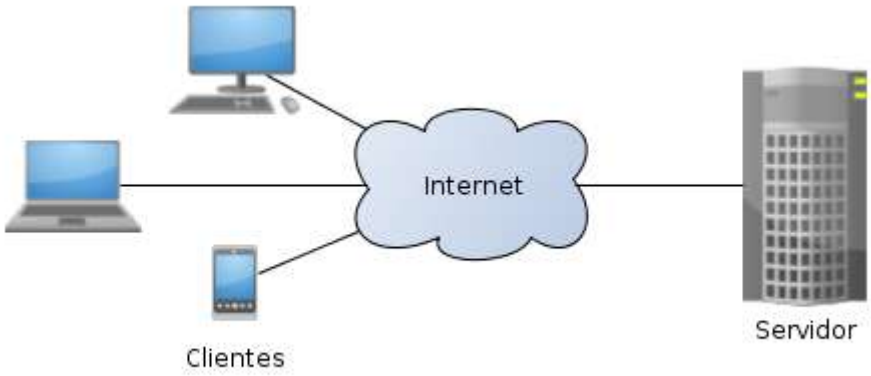


Figura 4.1: Modelo cliente-servidor [16]

Como se puede observar en la imagen, los clientes accederán a internet para solicitar al servidor la información requerida de la base de datos, esta conexión a internet con las distintas peticiones se llevará a cabo mediante llamadas a los distintos scripts PHP referentes a cada tipo de solicitud, se encargarán de recibir y enviar información para realizar aquellas acciones que los clientes necesiten operar sobre la base de datos para obtener o modificar información.

4.2 Base de datos

El primer requisito que se presenta al estudiar las distintas opciones para las bases de datos es que esta tiene que ser externa al dispositivo que actuará como cerradura. Teniendo esto en cuenta debemos crear una base de datos y colgarla en un servidor externo, para que tanto la aplicación de escritorio como el sistema que lea la identificación del usuario que desee acceder a una habitación puedan conectarse con ella.

Entre los diferentes tipos de base de datos [1], podemos encontrar los siguientes:

- MySql: es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.
- PostgreSql y Oracle: Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.
- Access: Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa Access, el cual crea un archivo .mdb con la estructura ya explicada.
- Microsoft SQL Server: es una base de datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.

Puesto que no vamos a trabajar con grandes volúmenes de información y cumple con todos los requisitos de nuestro sistema, además de tener una excelente accesibilidad desde todas las

plataformas que vamos a utilizar en el resto del proyecto, se desarrollará una base de datos MySQL.

A continuación se describirán las tablas que compondrán la base de datos:

4.2.1 Tabla clientes

#	Nombre	Tipo
1	ID_Cliente	int(3)
2	Nombre	varchar(32)
3	DNI	varchar(9)
4	Habitacion	int(3)
5	Telefono	int(9)

Figura 4.2: Tabla clientes

Como podemos ver en la figura 4.2, en la tabla clientes se encuentran todos los datos necesarios para definir a cada cliente y poder contactar convenientemente con él durante su estancia o con posterioridad. Posee los atributos:

- ID_Cliente: clave primaria de la tabla clientes e identificador único para cada cliente agregado a la base de datos.
- Nombre: contiene el nombre del cliente para referirse correctamente a él o ella.
- DNI: contiene el número del documento nacional de identidad de quien se registre, parte de la información necesaria para registrar correctamente al cliente.
- Habitacion: en este campo se agrega la habitación que se le haya asignado al cliente, necesario para contrastar por una parte que dicha habitación está libre, para cambiar la disponibilidad de la misma a ocupada y para permitir al cliente acceder a dicha habitación con la tarjeta que se le facilite.
- Telefono: contiene el número de teléfono del cliente, información orientada al contacto con el mismo.

4.2.2 Tabla empleados

#	Nombre	Tipo
1	ID_Empleado	int(3)
2	Nombre	varchar(32)
3	DNI	varchar(9)
4	Telefono	int(9)
5	Libertad_acceso	set('Total', 'Limpieza', 'Administracion')

Figura 4.3: Tabla empleados

Como podemos ver en la figura 4.3, en la tabla empleados se encuentran los datos necesarios para gestionar la información de cada trabajador a un nivel funcional desde el punto de vista del sistema que se está desarrollando, es decir, se agrega información de identificación, contacto y acceso a las distintas zonas del hotel. Sus atributos son los siguientes:

- ID_Empleado: clave primaria de la tabla clientes e identificador único para cada empleado agregado a la base de datos.
- Nombre: contiene el nombre del trabajador para referirse correctamente a él o ella.
- DNI: contiene el número del documento nacional de identidad del empleado, parte de la información de identificación de cada trabajador.
- Telefono: contiene el número de teléfono del empleado, información orientada al contacto con el mismo.
- Libertad_acceso: contiene la información necesaria para saber a que instancias del hotel puede acceder el trabajador en cuestión. Esta libertad de acceso puede ser Total, para aquel o aquellos trabajadores que deban tener la potestad de acceder en cualquier momento a cualquier habitación del hotel, de Limpieza, con la cual se podría acceder a las habitaciones de todas las plantas para poder realizar la limpieza de las mismas en los horarios específicos y por último Administración, para aquellos trabajadores que se encarguen de gestionar el funcionamiento del hotel y por consiguiente necesitarán acceder a las habitaciones restringidas con los equipos que lo permiten.

4.2.3 Tabla habitaciones

#	Nombre	Tipo
1	ID_Habitacion	int(3)
2	Disponibilidad	set('Libre', 'Ocupada', 'No_Disponible', '')
3	N_Camas	int(1)
4	Precio	int(4)

Figura 4.4: Tabla habitaciones

Tal y como se observa en la figura 4.4, en la tabla habitaciones encontramos la información necesaria para gestionar cada habitación, principalmente nos sirve para conocer su estado en cuanto a la disponibilidad de la misma y sus características. Posee los siguientes atributos:

- ID_Habitacion: clave primaria de la tabla habitaciones e identificador único para cada una. Coincide con el número de la habitación.
- Disponibilidad: contiene la disponibilidad de cada habitación, Libre u Ocupada en función de su ocupación o No Disponible para aquellas habitaciones que por reforma o algún incidente no se puedan utilizar.
- N_Camas: contiene el número de camas de la habitación, orientado a facilitar al personal encargado del registro de clientes la información necesaria que se ajuste a las necesidades del cliente.
- Precio: contiene el precio de la habitación, orientado a facilitar al personal encargado del registro de clientes la información necesaria que se ajuste a las necesidades del cliente.

4.3 Scripts PHP

Siguiendo la arquitectura del proyecto, para crear la comunicación entre los distintos clientes del proyecto con la base de datos será necesario un método disponible desde los dos entornos que manejarán los clientes, C# en el caso de la aplicación de escritorio y Python en el caso de las Raspberry encargadas de gestionar el control de acceso a las puertas del hotel. Un método sencillo de gestionar esta comunicación son los scripts PHP.

Los scripts PHP serán los encargados de gestionar las peticiones de los clientes, tanto la gestión de la información de la base de datos como la solicitud de códigos de acceso. Por esto la solución planteada consiste en generar un conjunto de archivos PHP para bajo la misma estructura básica realizar unas consultas u otras según las necesidades del cliente.

Las distintas necesidades en función de la petición de la parte cliente del sistema se pueden dividir en cinco grupos.

4.3.1 Scripts para mostrar información listada

Estos scripts serán los encargados de ejecutar una sentencia SELECT en su comunicación con la base de datos solicitando información a la tabla clientes, empleados o habitaciones.

Estos scripts serán **Lista_clientes.php**, que solicitará la información del ID del cliente, su nombre y su habitación para cualquier gestión relacionada con el mismo. Esto nos permitirá mostrar esta información en una tabla que muestre a todos los clientes que tendrá registrados en ese momento el hotel. De igual modo hará el script **Lista_empleados.php**, que solicitará información de ID del empleado, su nombre y libertad de acceso. Por último tenemos el script **Lista_habitaciones.php**, que solicitará toda la información de las habitaciones del hotel, el ID de la habitación, la disponibilidad, el número de camas y el precio por habitación. Este script nos permitirá mostrar una lista con toda la información de las habitaciones del hotel actualizada en todo momento.

4.3.2 Scripts para eliminar información

Estos scripts serán los encargados de ejecutar una sentencia DELETE en su comunicación con la base de datos, dando así de baja cualquier registro de la tabla clientes, empleados o habitaciones.

Los scripts serán **Baja_clientes.php**, que eliminará al cliente seleccionado una vez abandone el hotel, actualizando así el listado de los clientes en ese momento. De igual modo los scripts **Baja_empleados.php** y **Baja_habitaciones.php** nos permitirán dar de baja a un empleado o una habitación según la circunstancia.

4.3.3 Scripts para agregar información

Estos scripts serán los encargados de ejecutar una sentencia INSERT en su comunicación con la base de datos, agregando así un nuevo registro de la tabla clientes, empleados o habitaciones.

Los scripts serán **Nuevo_cliente.php**, que agregará un nuevo cliente a la tabla clientes con todos sus campos correspondientes de la base de datos, ID_Cliente, Nombre, DNI, Habitacion y Telefono. El script **Nuevo_empleado.php** agregará un nuevo empleado en la tabla clientes con todos sus campos, ID_Empleado, Nombre, DNI, Telefono y Libertad_acceso. Por último, el script **Nueva_habitacion.php** nos permitirán agregar una nueva habitación al sistema de gestión del hotel en el caso de querer controlar el acceso en alguna estancia que no se considerase en un primer momento o de poseer más habitaciones disponibles. Agregará un nuevo registro en la tabla habitaciones con todos sus campos, ID_Habitacion, Nombre_Habitacion, Disponibilidad, N_Camas, Precio, Tarjeta_habitacion y Acceso_requerido.

4.3.4 Scripts para modificar información

Estos scripts serán los encargados de ejecutar una sentencia UPDATE en su comunicación con la base de datos, permitiendo así modificar cualquier registro de la tabla clientes, empleados o habitaciones.

En este caso los tres scripts funcionarán exactamente de la misma manera, variando únicamente en los campos a modificar en función de la tabla con la que se quiera trabajar. Los scripts son:

Modificaciones_clientes.php, **Modificaciones_empleados.php** y **Modificaciones_habitaciones.php**, que nos permitirán modificar cualquier campo de un registro en función de las necesidades del usuario de la aplicación de escritorio.

4.3.5 Scripts para control de acceso

Por último, estos scripts serán los encargados de enviar información de códigos de acceso de la base de datos a las cerraduras, permitiendo de este modo comparar la información y comprobar si se debe permitir el acceso. Estos scripts serán: **ID-Libertad_Acceso.php, Habitacion-Tarjeta.php** y **Habitacion-Libertad_Acceso.php,** los cuales nos permitirán obtener y comparar la distinta información según la tarjeta detectada por el lector de la cerradura.

4.4 Diseño de la aplicación de escritorio

Para el desarrollo de la aplicación de escritorio partimos de que se va a desarrollar con Microsoft Visual Studio, dado que permite generar fácilmente aplicaciones con interfaz gráfica y posee un gran catálogo de bibliotecas de clases que nos permitirán interactuar con todas las demás plataformas utilizadas en el proyecto.

Sin embargo, dentro de Visual Studio podemos programar nuestra aplicación en tres entornos de desarrollo diferentes: Visual Basic, Visual C# y visual C++.

Visual Basic [10] está diseñado para crear de manera productiva aplicaciones con seguridad de tipos orientadas a objetos. Visual Basic permite a los desarrolladores establecer como destino dispositivos móviles, web y Windows. Al igual que todos los lenguajes que tienen como destino Microsoft .NET Framework, los programas escritos en Visual Basic se benefician de la seguridad y la interoperabilidad entre lenguajes.

Visual C++ [11] es un entorno de desarrollo integrado (IDE) para lenguajes de programación C, C++ y C++/CLI.

Visual C++ engloba el desarrollo de aplicaciones hechas en C, C++ y C++/CLI en el entorno Windows. Visual C++ incluye además las bibliotecas de Windows (WinApi), las bibliotecas MFC y el entorno de desarrollo para .NET Framework. Visual C++ cuenta con su propio compilador (de igual nombre). Además provee de bibliotecas propias de cada versión del sistema operativo y sockets.

El lenguaje de programación utilizado por esta herramienta, de igual nombre, está basado en C++ y es compatible en la mayor parte de su código con este lenguaje, a la vez que su sintaxis es exactamente igual. En algunas ocasiones esta incompatibilidad impide que otros compiladores, sobre todo en otros sistemas operativos, funcionen bien con código desarrollado en este lenguaje.

Visual C# [12] es un nuevo entorno de programación diseñado para crear un amplio número de aplicaciones empresariales que se ejecutan en .NET Framework. Es sencillo, moderno, proporciona seguridad de tipos y está orientado a objetos. El código creado mediante C# se compila como código administrado, lo cual significa que se beneficia de los servicios de Common Language Runtime. Estos servicios incluyen interoperabilidad entre lenguajes,

recolección de elementos no utilizados, mejora de la seguridad y mayor compatibilidad entre versiones.

Puesto que las tres opciones cumplen con los requisitos de nuestra aplicación, y todas ellas tienen herramientas que permiten la compatibilidad con el resto de plataformas utilizadas en el proyecto, se desarrollará la aplicación con Visual C# debido a que es un entorno más similar a C que Visual Basic y más sencillo que Visual C++ a la hora de crear entonos gráficos sin experiencia previa en el uso de plataformas .NET.

4.4.1 Diseño de pantallas

En este apartado se mostrará una primera versión de la distribución de pantallas de la aplicación de escritorio anterior al periodo de desarrollo.

El primer prototipo mostrado en la figura 4.5, muestra una distribución de la aplicación que separa las pantallas en 3 grupos, clientes, habitaciones y empleados, pasando posteriormente al registro, baja o modificaciones en el caso de clientes y empleados y a una ventana que muestra el listado de las habitaciones del hotel o a otra que permite deshabilitar una habitación en caso de que no esté disponible para reserva.

En el segundo prototipo mostrado en la figura 4.6 se observa que la principal diferencia es la agregación de botones para volver a la ventana anterior, también se unifican todas las funciones de la ventana habitaciones en una sola ventana y se suprimen las distintas opciones de modificar información de los clientes dejando únicamente una ventana para ello.

4.4.1.1 Primer prototipo

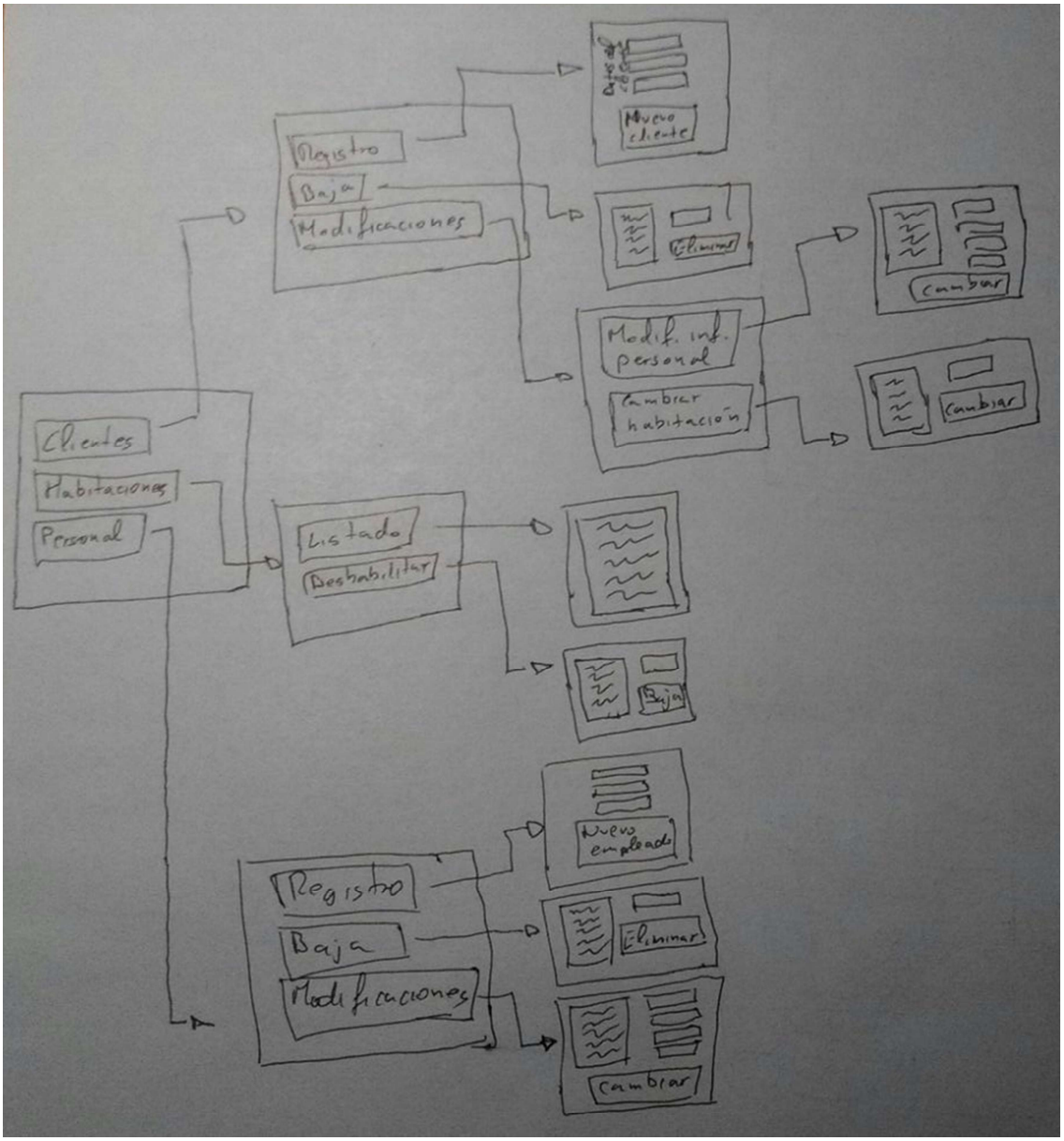


Figura 4.5: Prototipo inicial de pantallas

4.4.1.2 Segundo prototipo

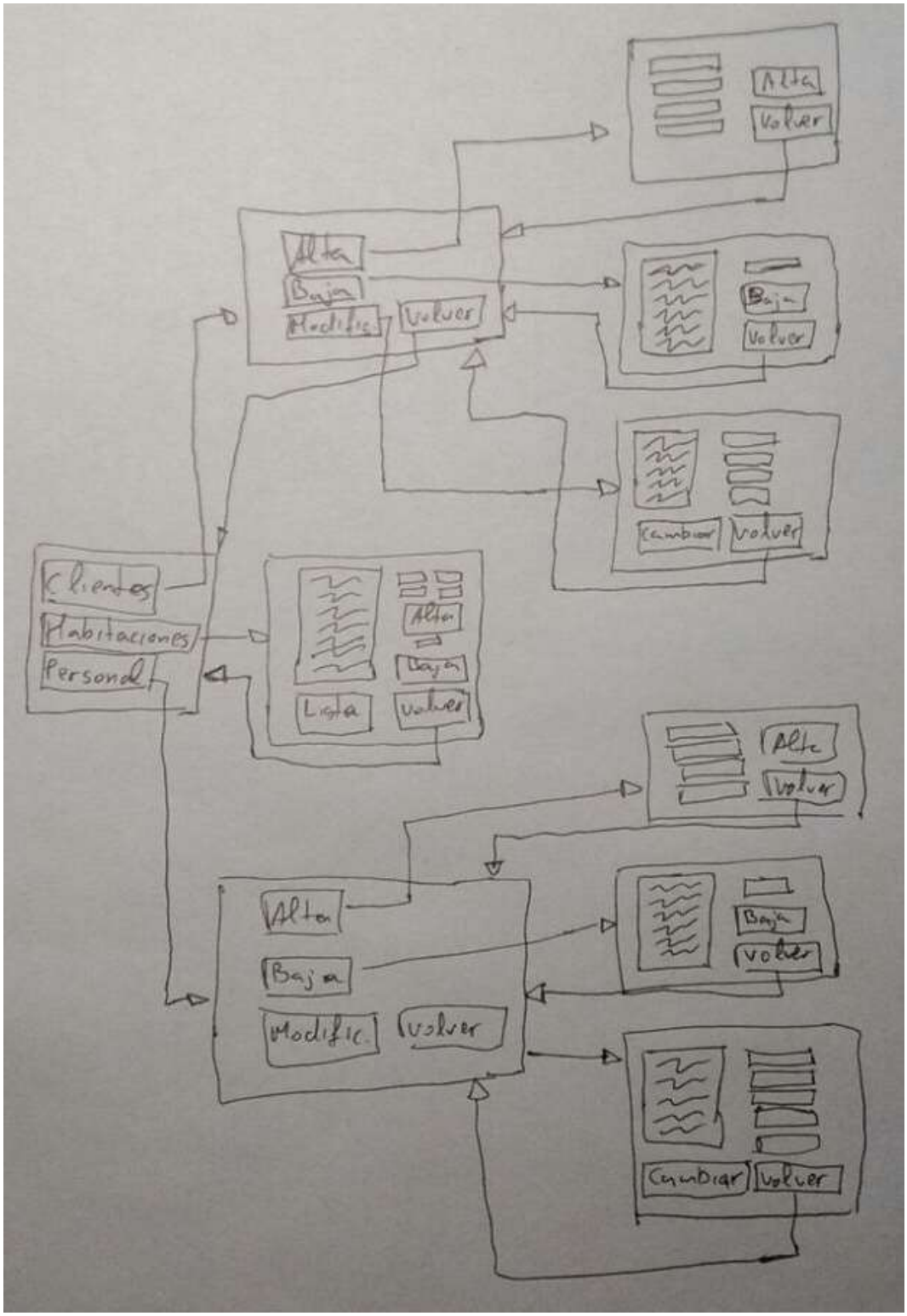


Figura 4.6: Segundo prototipo de pantallas

4.5 Servidor

Siguiendo el modelo cliente servidor debemos tener acceso a un servidor en el cual alojar nuestra base de datos. Este alojamiento de la base de datos y la llamada a los scripts PHP los diseñaremos siguiendo dos fases de desarrollo en función de las necesidades de cada una.

4.5.1 Fase inicial de desarrollo

En esta fase inicial trabajaremos simultáneamente con la aplicación de escritorio, los scripts PHP y la base de datos, por tanto, utilizaremos una base de datos local alojada en el equipo de desarrollo y a la cual poder acceder rápidamente y sin necesidad de conexión a internet.

Para esto elegiremos el servidor independiente XAMPP, ya que contiene la base de datos MySQL e intérpretes para lenguajes de script PHP. El motivo principal por lo que se selecciona MySQL es porque es un servidor de muy fácil instalación, así como que es un software libre y por tanto gratuito, tanto el programa como todas sus extensiones.

4.5.2 Fase final de desarrollo

Una vez el proyecto esté en una fase más avanzada, y tengamos que procurar una comunicación entre el dispositivo de control de acceso y la base de datos, no podremos hacerlo llamando a nuestra base de datos alojada en el equipo de desarrollo de la aplicación de escritorio, ya que desde el dispositivo de control de acceso no podrá acceder a la misma.

Para esto deberemos cambiar la ruta a la que apunten los scripts PHP del localhost utilizado para acceder a la información que se aloja en nuestro servidor XAMPP de forma interna en el equipo de desarrollo de la aplicación de escritorio por una dirección web en la que tendremos nuestra base de datos. Puesto que XAMPP tiene el servidor web Apache, esto nos será posible cambiando la dirección desde la que se ejecutarán nuestros scripts por la dirección IP del equipo en el que se encuentra instalado XAMPP y que tiene los scripts PHP, típicamente del tipo 192.168.1.XXX. Esto nos permitirá ejecutar los scripts de forma remota de una forma exactamente igual a la que lo haríamos si contratásemos un dominio web en el cual alojar los scripts y la base de datos. Esto será posible debido a que tanto el equipo en el cual tenemos instalado el servidor XAMPP, como la Raspberry que realizará las llamadas desde un punto externo, estarán conectadas a la misma red; permitiendo así un acceso a través de ella.

4.6 Sistema de reconocimiento de identificación

La identificación en el punto de acceso se puede realizar de diversas formas, para este proyecto hemos considerado dos: identificación mediante código PIN insertado por teclado alfanumérico o identificación mediante tarjetas sin contacto RFID.

Para el sistema propuesto, el control de acceso de las puertas de un hotel, sería bastante útil proporcionar un PIN único a cada empleado para acceder a las diferentes instancias del hotel, sin embargo, no parece una buena idea utilizar para reconocer la identificación de usuario un teclado alfanumérico en cada una de las puertas, puesto que habría que asignar un PIN a cada cliente que se puede olvidar o extraviar fácilmente. Por tanto, se utilizará un lector para tarjetas

sin contacto RFID en todas las puertas del hotel, y este sistema será el encargado de reconocer al usuario y gestionar su acceso.

Este lector de tarjetas sin contacto será el módulo lector RFID RC522, debido a su bajo precio y fácil conexión mediante SPI.

4.7 Sistema de control de acceso

Para el diseño del sistema empotrado encargado de gestionar el control de acceso en nuestro proyecto, primero debemos estudiar las tres alternativas más extendidas en este momento en el mercado para la parte computacional: Arduino, BeagleBone y Raspberry Pi.

Arduino [13] es una plataforma de prototipos electrónica de código abierto basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede interactuar con el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador.

BeagleBone [14] es un ordenador pequeño del tamaño de una tarjeta de crédito, donde puedes ejecutar un sistema operativo, como puede ser Linux/Android 4.0. Su principal diferencia con Arduino es que puede ejecutar un pequeño sistema operativo, es prácticamente un miniordenador donde se pueden ejecutar programas sobre estos sistemas operativos. BeagleBone, está diseñado para funcionar a un nivel mucho más alto y tiene mucha más capacidad de proceso que Arduino.

Raspberry Pi es un miniordenador con el tamaño de una tarjeta de crédito, bastante barato (más que BeagleBone, pero menos que Arduino), desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo principal de estimular la enseñanza de ciencias de la computación en las escuelas. Al ser un miniordenador podemos utilizarlo para desarrollar cosas bastante más complejas que con Arduino y puede utilizar lenguajes de programación de alto nivel como Python, C++ y Java.

Frente a estas tres alternativas hemos optado por Raspberry Pi debido a que posee mayor potencia de procesamiento que Arduino, es más adecuada para realizar proyectos que requieran el uso de una pantalla o conectividad de red y tiene una excelente relación capacidades-precio.

También se ha tenido en cuenta el interés por el alumno en aprender a desenvolverse en el entorno de Raspberry Pi y la disposición de una en el laboratorio GUTI de la universidad.

4.7.1 Diseño del sistema de control de acceso

Una vez hemos aclarado la elección de la placa que vamos a utilizar a modo de sistema empotrado en las puertas del hotel, vamos a elaborar un resumen con los elementos de los que constará dicho sistema.

- Computador: la función de computación, como se acaba de especificar, se realizará mediante el uso de una Raspberry Pi, en concreto una Raspberry Pi 3 Model B V1.2.

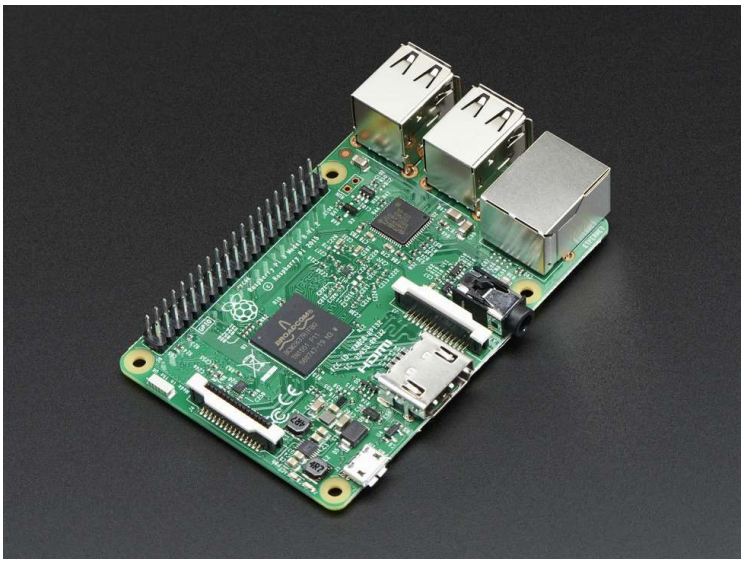


Figura 4.7: Raspberry Pi 3 modelo B [17]

- Pantalla: Para mostrar al usuario que tras el reconocimiento de su tarjeta de acceso tiene permiso o no para abrir la puerta en cuestión, se utilizará un display LCD de 2.2" que se comunicará mediante SPI a la Raspberry.

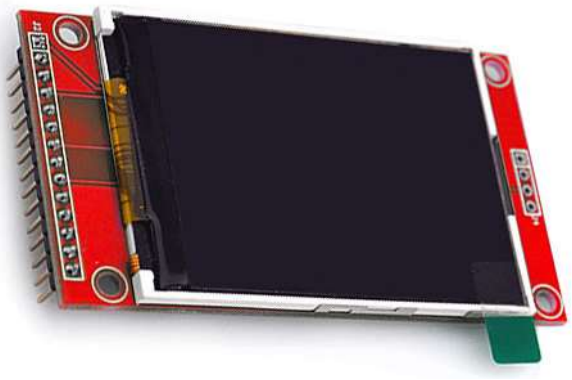


Figura 4.8: Display LCD 2.2" [18]

- Sistema de reconocimiento de identificación: como ya se ha especificado en el capítulo 4.7, el reconocimiento de identificación se realizará mediante el módulo lector RFID RC522, que se conectará mediante el protocolo SPI directamente a la Raspberry, permitiendo de este modo reconocer una tarjeta en cualquier momento e iniciar la comprobación de acceso del usuario a la instancia en cuestión.

Figura 4.9: Módulo RFID con tarjeta [18]

- Hat-Board: para poder conectar de una forma sencilla mediante comunicación SPI tanto la pantalla como el sistema de reconocimiento de identificación con la Raspberry, se utilizará una Hat-Board. Una Hat-Board (Hardware Attached on Top o Hardware Adjunto en la parte Superior) es una placa rectangular compatible con el modelo B+ de Raspberry que permite al usuario conectar una gran variedad de complementos en su computador se forma sencilla. Cuando se conecta la Hat-Board a la Raspberry, ésta la reconoce y configura automáticamente los GPIOs y controladores para la placa, simplificando enormemente su uso. Para el desarrollo de este proyecto hemos seleccionado la Hat-Board IRP-102.

Figura 4.10: Hat-Board iRP-102 [18]

De esta forma tendríamos una configuración del hardware del sistema que seguiría el siguiente esquema:

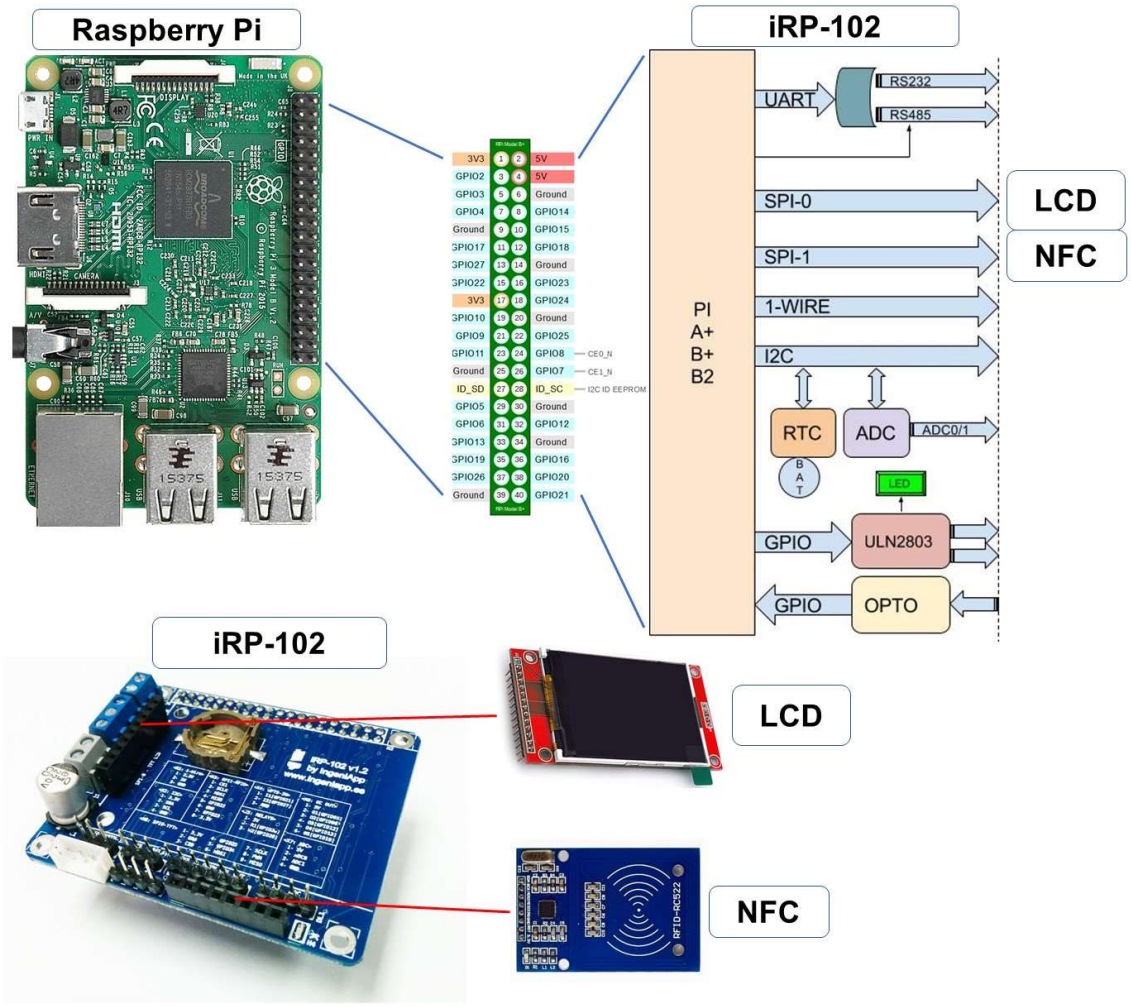


Figura 4.11 Esquema de conexión Raspberry-Hat board-lector RFID-display LCD [18][19][20]

En la parte superior de la Figura 4.11 se puede ver como la Hat-Board iRP102 se conecta a los 40 pines GPIO de la Raspberry Pi, conectando así el display LCD y el lector RFID en el bus SPI-0 y el puerto SPI-1 respectivamente. En la parte inferior se puede observar cómo se conectarán tanto el display como el lector RFID en la Hat-Board de una forma física.

5 Desarrollo del sistema

En este capítulo se expondrá el desarrollo del sistema completo, desde la instalación del software hasta el desarrollo del código necesario para conseguir un correcto funcionamiento del sistema planteado, así como todas las configuraciones de hardware y software requeridas y la estructura de directorios de los archivos que requieran una localización específica para su correcto funcionamiento o relación con el resto de elementos del sistema.

5.1 Desarrollo de la base de datos

Como ya se ha especificado en el apartado de análisis y diseño del sistema, la base de datos se desarrollará en SQL. En un principio estará alojada en un servidor interno instalado en el propio equipo, y posteriormente se alojará en un servidor externo con el fin de acceder a ella desde el resto de clientes externos al equipo de la aplicación de escritorio, los sistemas de control de acceso compuestos por una Raspberry Pi, un lector RFID y un display LCD.

5.1.1 Servidor XAMPP

Se ha utilizado como servidor interno en el equipo el servidor independiente multiplataforma XAMPP, a fin de facilitar el desarrollo y las pruebas de la base de datos antes de alojarla en un servidor externo cuando se desarrolle la parte cliente del sistema.

A continuación, mostraremos una breve explicación del manejo del panel de control.

Podemos ver el panel de control en la figura 5.1.

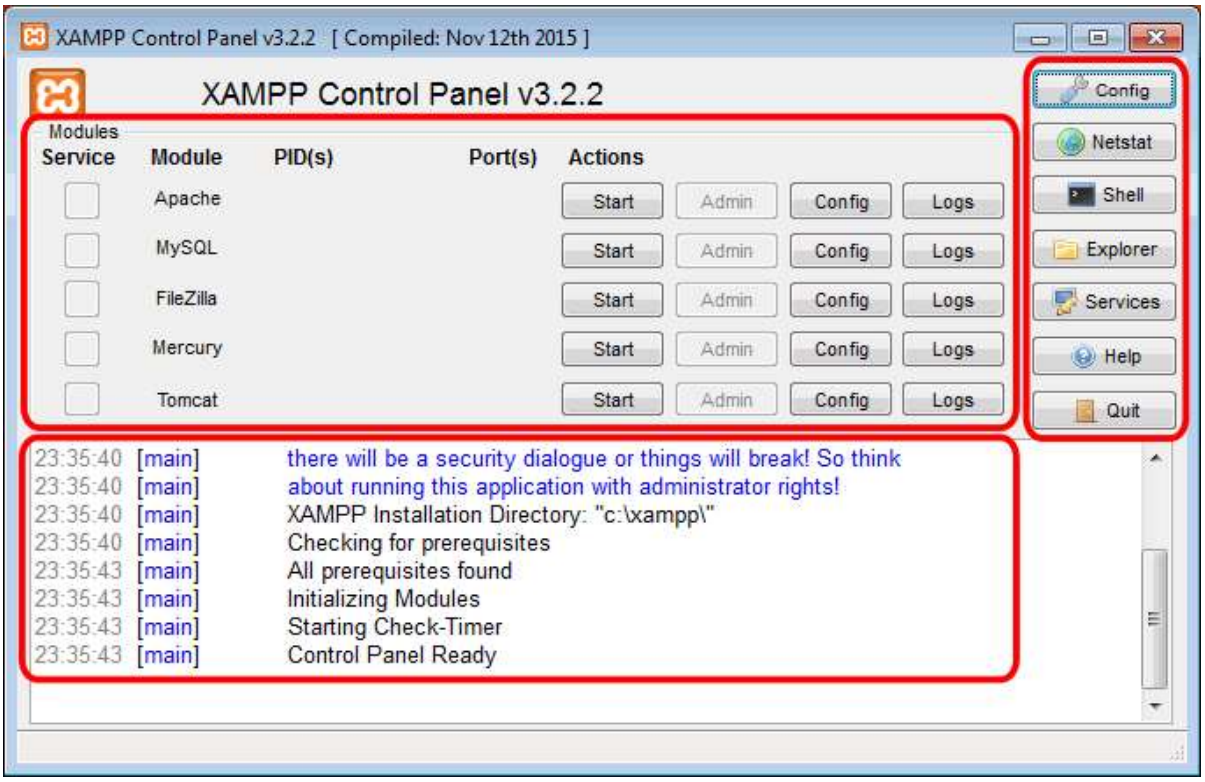


Figura 5.1: Panel de control XAMPP [15]

La parte superior es la zona de módulos, en ella podemos ver para cada uno de los módulos de XAMPP: si está instalado como servicio, su nombre, el identificador de proceso y el puerto utilizado, así como botones para iniciar y detener los procesos, administrarlos, editar los archivos de configuración y abrir los archivos de registro de actividad respectivamente.

La zona de notificaciones en la parte inferior XAMPP informa del éxito o fracaso de las acciones realizadas.

Por último, a la derecha tenemos la zona de utilidades, para acceder rápidamente a la configuración de XAMPP, un listado de las conexiones activas del equipo, la terminal de XAMPP, la carpeta de instalación, un acceso a los servicios locales del equipo, una interfaz de ayuda con información de uso de XAMPP y el botón "Quit" que será la única forma de cerrar XAMPP, ya que si lo hacemos desde la cruz de cerrar que se encuentra en la esquina superior derecha, el panel de control se ocultará, pero el servidor seguirá activo.

Con nuestro servidor instalado procederemos al último paso, configurar el módulo Apache y MySQL como servicios, permitiendo de este modo que se pongan en marcha cada vez que arrancamos el ordenador.

Con el servidor instalado y configurado ya podemos acceder a la dirección http://localhost/phpmyadmin/ desde nuestro navegador para acceder a la herramienta phpMyAdmin.

El acceso a los scripts PHP necesarios para la comunicación entre servidor y clientes serán llamados introduciendo la dirección "localhost/archivo.php", permitiendo así a las distintas funciones de la aplicación de escritorio, sin embargo, una vez terminado el desarrollo de la

aplicación de escritorio, para llamar a los scrips desde la Raspberry actuando a modo de cliente externo al servidor, llamaremos a estos scripts PHP introduciendo la dirección IP del equipo en el cual instalamos XAMPP seguida por el puerto correspondiente separado por dos puntos. Esto es posible debido a que XAMPP integra el servidor web Apache, el cual aloja los archivos PHP en su carpeta htdocs y permite ejecutarlos como en una llamada a una web externa.

Como ejemplo clarificador, se presentará a continuación un ejemplo de llamada a un archivo PHP desde la Raspberry realizada durante el desarrollo del sistema:

url: 192.168.1.105:80/nombre_del_archivo.php

Apache utiliza los puertos 80 y 443, por tanto cualquiera de ellos podrá utilizarse para llamar a los archivos alojados en dicho servidor.

5.1.2 Base de datos

La base de datos se desarrollará mediante el uso de phpMyAdmin, una herramienta escrita en PHP destinada a facilitar la creación, modificación y el manejo de las bases de datos MySQL a través de una interfaz gráfica de usuario. Se encuentra integrado en la mayoría de alojamientos que ofrezcan servicios de hosting de bases de datos MySQL.

Una vez nos encontremos en la interfaz de phpMyAdmin lo primero será crear nuestra nueva base de datos. Para ello pulsaremos el botón “Nueva” que se encuentra en el lado izquierdo de la pantalla tal y como se muestra en la figura 5.2:

Figura 5.2: Crear nueva base de datos

Ahora tendremos que introducir el nombre de nuestra nueva base de datos, así como su cotejamiento, que estableceremos en “utf8_spanish_ci” que corresponde con lenguaje español moderno.

Con nuestra nueva base de datos creada procederemos a crear las tablas de la misma, para nuestro proyecto tendremos un total de 3 tablas: clientes habitaciones y empleados. Para ello introduciremos el nombre de la tabla y el número de columnas tal y como se muestra en la siguiente imagen:

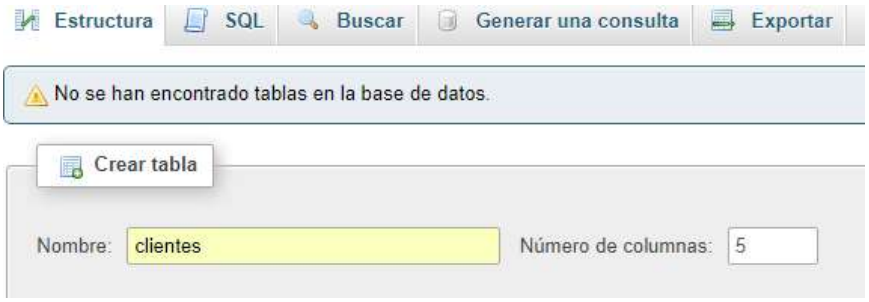


Figura 5.3: Crear tabla

Del mimso modo generamos las otras 2 tablas, empleados con 6 columnas y habitaciones con 7, obteniendo así nuestra base de datos con sus tablas y campos listos para almacenar información pero sin ningún registro en ellas:

Figura 5.4: Tablas creadas en la base de datos

Por último, introduciremos los registros en cada una de las tablas, para ello seleccionaremos la tabla a la que queramos agregar información y pulsaremos “Insertar” en el menú superior:

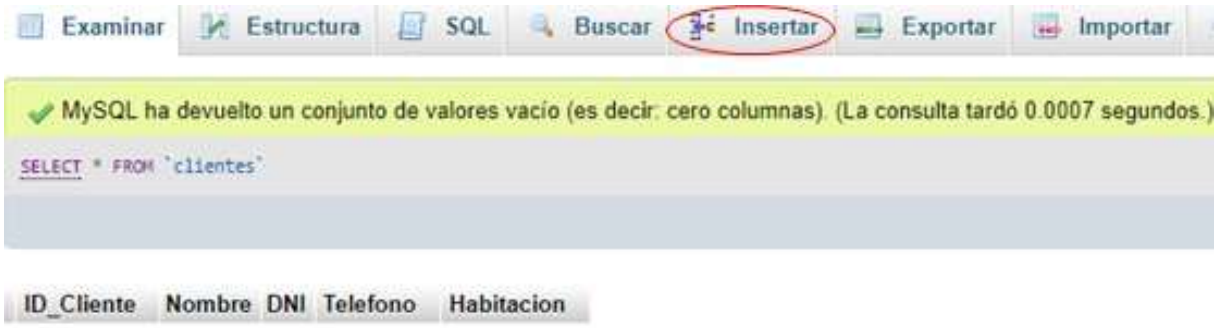


Figura 5.5: Acceder a inserción de datos

Una vez en el menú insertar, podremos agregar tantos registros como queramos, modificando la opción que aparece en la parte inferior, por debajo de los 2 registros que te da la opción de crear por defecto, comenzaremos la inserción con dos clientes como ejemplo:

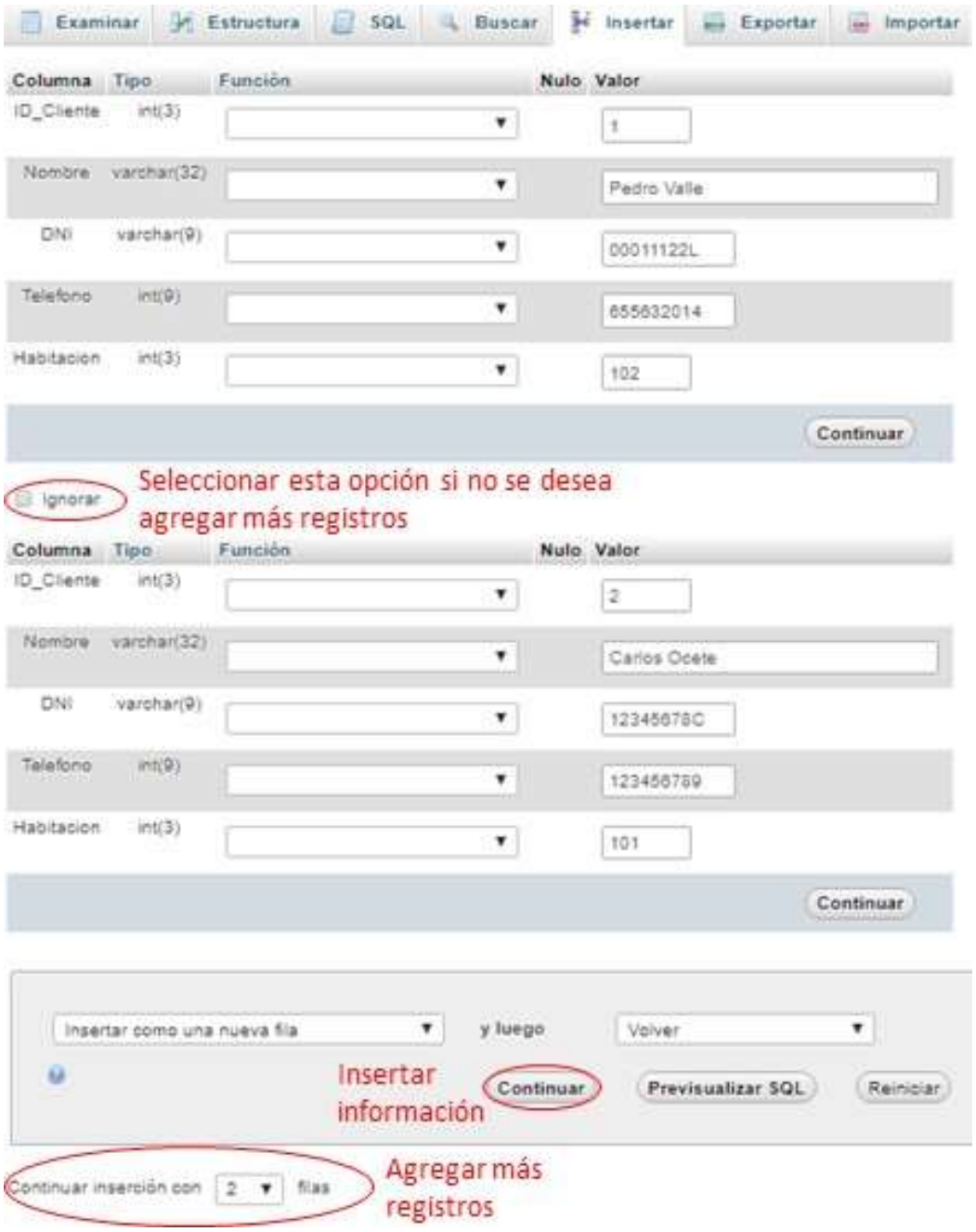


Figura 5.6: Insertar datos en las tablas

Seguimos el mismo procedimiento con la tabla de empleados y la tabla de habitaciones, introduciremos 3 empleados con cada una de las libertades de acceso, con el fin de poder hacer posteriormente las pruebas necesarias sobre su interacción con el resto de elementos del sistema y un total de 20 habitaciones, 15 de ellas dormitorios divididos en 3 plantas y las otras 5 de carácter administrativo o de uso del personal.

Una vez agregados todos los registros en nuestras tablas, para comprobar que están todos correctamente introducidos, seleccionaremos una tabla e iremos a la pestaña “Examinar” del

menú superior, obteniendo así toda la información presente en cada una de las tablas de la siguiente manera:

- Tabla clientes:

	ID_Cliente	Nombre	DNI	Habitacion	Telefono
Editar Copiar Borrar	1	Pedro Valle	00011122L	102	915648722
Editar Copiar Borrar	2	Carlos Ocete	12345678C	101	123456789

Seleccionar todo *Para los elementos que están marcados:* Editar Copiar Borrar Exportar

Figura 5.7: Tabla clientes con los registros insertados

- Tabla empleados:

Como se puede observar en la figura 5.8, se ha añadido el campo "Tag_empleado" que no estaba en el diseño de la tabla empleados mostrado en el punto 4.2.2, esta columna se añadió por cuestiones de funcionalidad durante el desarrollo del proyecto al considerar que era necesaria para el correcto funcionamiento del mismo.

	ID_Empleado	Nombre	DNI	Telefono	Libertad_acceso	Tag_empleado
Editar Copiar Borrar	1	Carlos Muñoz	70500111M	918559656	Total	21564905142
Editar Copiar Borrar	2	Ernesto Cadiz	15987426L	698532147	Administracion	61542198520
Editar Copiar Borrar	3	Alberto Sanz	56948137S	626887517	Limpieza	37138217101

Seleccionar todo *Para los elementos que están marcados:* Editar Copiar Borrar Exportar

Figura 5.8: Tabla empleados con los registros insertados

- Tabla habitaciones:

Podemos ver en la figura 5.9 que se han añadido los campos "Nombre_Habitacion", "Tarjeta_habitacion" y "Acceso_requerido", que no estaban en el diseño de la tabla habitaciones mostrado en el punto 4.2.3, estas columnas se añadieron por cuestiones de funcionalidad durante el desarrollo del proyecto al considerar que eran necesarias para el correcto funcionamiento del mismo.

	ID_Habitacion	Nombre_Habitacion	Disponibilidad	N_Camas	Precio	Tarjeta_habitacion	Acceso_requerido
Editar Copiar Borrar	1	Oficina del director	No_Disponible	0	0	0	Total
Editar Copiar Borrar	2	Comedor de personal	No_Disponible	0	0	0	Administracion
Editar Copiar Borrar	3	Back Office / recpecion	No_Disponible	0	0	0	Administracion
Editar Copiar Borrar	4	Oficina RRHH	No_Disponible	0	0	0	Administracion
Editar Copiar Borrar	5	Lavanderia	No_Disponible	0	0	2460135928	Limpieza
Editar Copiar Borrar	101	Dormitorio	Ocupada	2	100	0	Limpieza
Editar Copiar Borrar	102	Dormitorio	Ocupada	2	120	0	Limpieza
Editar Copiar Borrar	103	Dormitorio	Libre	2	100	2365941201	Limpieza
Editar Copiar Borrar	104	Dormitorio	Libre	2	110	2540249101	Limpieza
Editar Copiar Borrar	105	Dormitorio	Libre	2	120	0	Limpieza
Editar Copiar Borrar	201	Dormitorio	Libre	3	150	0	Limpieza
Editar Copiar Borrar	202	Dormitorio	Libre	3	140	0	Limpieza
Editar Copiar Borrar	203	Dormitorio	Libre	3	160	0	Limpieza
Editar Copiar Borrar	204	Dormitorio	Libre	3	150	0	Limpieza
Editar Copiar Borrar	205	Dormitorio	Libre	3	160	0	Limpieza
Editar Copiar Borrar	301	Dormitorio	Libre	1	70	0	Limpieza
Editar Copiar Borrar	302	Dormitorio	Libre	1	50	0	Limpieza
Editar Copiar Borrar	303	Dormitorio	Libre	1	60	0	Limpieza
Editar Copiar Borrar	304	Dormitorio	No_Disponible	1	50	0	Limpieza
Editar Copiar Borrar	305	Dormitorio	Libre	1	40	0	Limpieza

Seleccionar todo *Para los elementos que están marcados:* Editar Copiar Borrar Exportar

Figura 5.9: Tabla habitación con los registros insertados

Se puede observar que se han agregado columnas en la tabla empleados y la tabla habitaciones respecto a las propuestas en la sección de diseño, esto es debido a decisiones de desarrollo y estética que se tomaron posteriormente.

En la tabla empleados se ha agregado la columna "Tag_empleado" con el fin de facilitar el control de acceso de los mismos a las distintas habitaciones, accediendo mediante este código del tag a la libertad de acceso que posee dicho empleado.

En la tabla se habitaciones se han agregado varias columnas, "Nombre_Habitacion" que proporciona una breve descripción de cada estancia para facilitar el control de las mismas, evitando así que el empleado encargado de manejar toda la información del hotel tenga que saber a que habitación pertenece cada ID. Otra columna agregada respecto al diseño inicial es "Tarjeta_habitacion", que facilita el chequeo de acceso a las habitaciones por parte de los clientes, permitiendo comparar directamente el ID de la tarjeta que solicita el acceso a una estancia con el código que autoriza la apertura de la puerta en cuestión. Por último, se ha agregado la columna "Acceso_requerido" también con el fin de simplificar el control de acceso de los empleados a las distintas habitaciones del hotel, permitiendo así comparar la libertad de acceso asociada a un tag que solicita el acceso con la restricción de acceso de la puerta en la que se haya realizado la petición.

5.2 Desarrollo de la aplicación de escritorio

Como se aclaró en el capítulo de análisis y diseño del sistema, el desarrollo de la aplicación de escritorio se llevará a cabo mediante el uso de la aplicación Windows Forms de Visual Studio en el lenguaje C#.

El desarrollo de aplicaciones mediante esta herramienta consiste en la creación de ventanas o "Forms" en los que posteriormente podremos incluir una gran variedad de elementos y programar sus rutinas.

En el desarrollo de esta aplicación se han generado un total de 11 Forms. A continuación se mostrará un listado de las distintas ventanas que componen la aplicación con una imagen de ellas junto con una breve descripción de sus elementos así como su funcionamiento.

- **Inicio:**

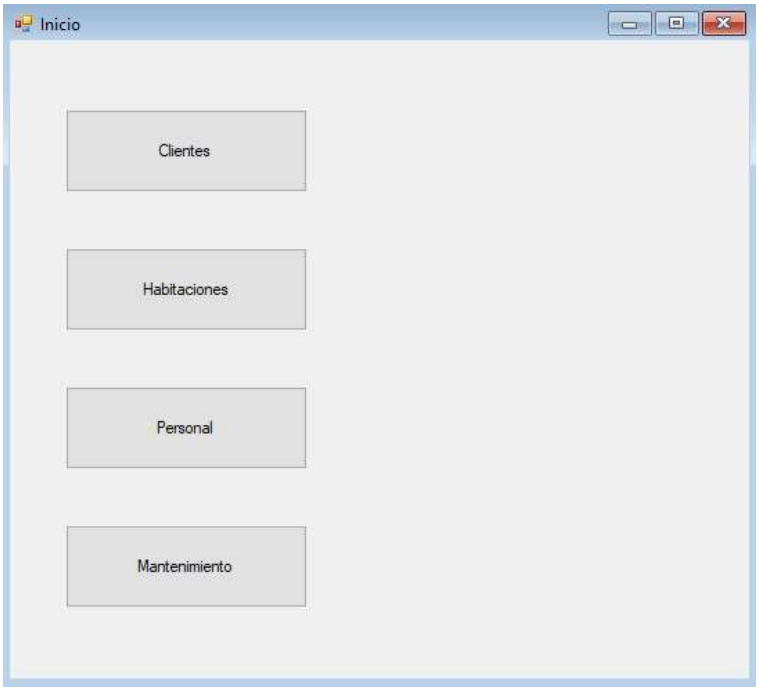


Figura 5.10: Ventana Inicio

en esta ventana tendremos un primer menú que permitirá al empleado encargado del manejo de la aplicación de escritorio manejar la información de cualquiera de las 3 tablas disponibles: clientes, empleados y habitaciones.

En la figura 5.10 podemos observar que la ventana de Inicio consta de 3 botones, cada uno de ellos encargado de cerrar la ventana actual y mostrar la indicada en estos botones: Clientes, Habitaciones y Personal.

- **Clientes:**

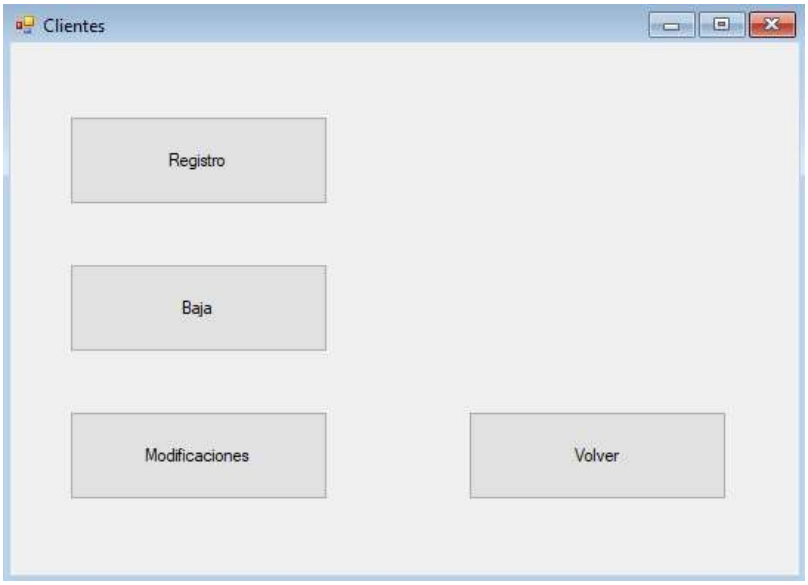


Figura 5.11: Ventana Clientes

En esta ventana accederemos a un menú desde el que podremos gestionar la información de los clientes del hotel o volver a la ventana de inicio.

En la figura 5.11 podemos observar que la ventana de Clientes se compone de 4 botones, 3 de ellos encargados de cerrar la ventana actual y abrir cualquiera de las 3 ventanas que gestionan la información de la tabla clientes: Registro, Baja y Modificaciones. El cuarto botón es el de Volver, que cierra la ventana actual y abre la ventana superior, en este caso la ventana Inicio.

- **Registro clientes:**

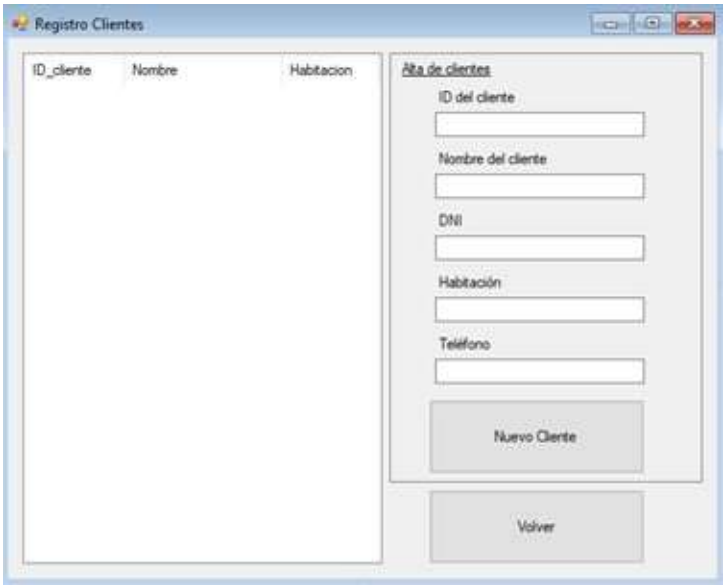


Figura 5.12: Ventana Registro clientes

En esta ventana se facilitará la información necesaria para ingresar un nuevo cliente en la base de datos del hotel, así como objetos para recoger la información requerida para esto que introducirá el usuario.

Como se puede observar en la figura 5.12, esta ventana consta de un objeto ListView, que mostrará el ID, nombre y habitación en la que se encuentran registrados los clientes actuales en la base de datos del hotel para facilitar así al empleado la información que necesite. También consta de 5 TextBox con sus respectivos Label en la zona superior de estos para indicar la información que el empleado deberá introducir en los TextBox para registrar al nuevo cliente. Por último, tiene dos botones, el de Volver que cierra la ventana actual y muestra la ventana superior, en este caso la ventana Clientes, y el botón Nuevo Cliente, que introducirá un nuevo cliente en la base de datos si la información requerida se ha introducido correctamente en los ListBox, siguiendo el funcionamiento representado en la figura 5.25.

- **Baja clientes:**

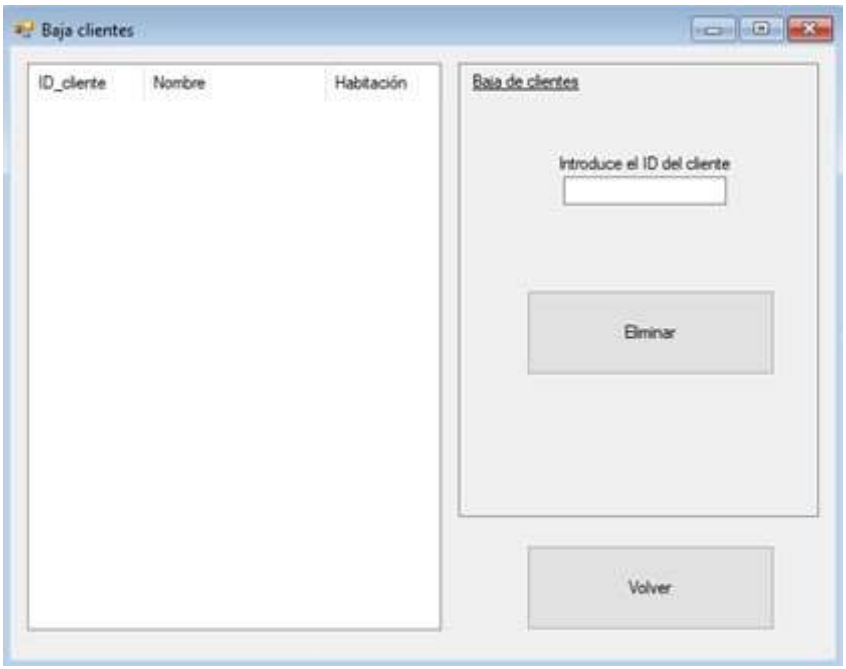


Figura 5.13: Ventana Baja clientes

En esta ventana el empleado encargado de manejar la aplicación de escritorio podrá ver la información necesaria para dar de baja a un cliente y llevar a cabo la operación.

Como se puede observar en la figura 5.13 esta ventana está compuesta por un ListView para poder ver la información que deberá introducir en los dos TextBox que se encuentran acompañados de sus respectivos Labels. Por último tenemos dos botones, el de Volver que cierra la ventana actual y muestra la ventana superior, en este caso la ventana Clientes, y el botón Eliminar, que elimina al cliente introducido por el empleado si la información ha sido introducida correctamente y cambiará la columna

Disponibilidad de la habitación que tenía asignada este cliente, poniéndola en Libre, siguiendo el funcionamiento representado en la figura 5.27.

- **Modificaciones clientes:**

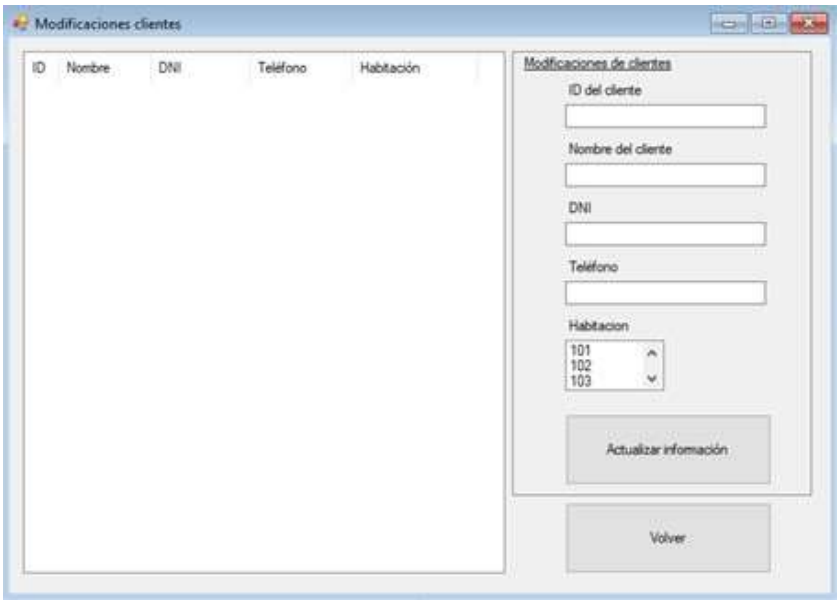


Figura 5.14: Ventana Modificaciones clientes

Esta ventana permitirá al empleado modificar la información de cualquier columna de los registros actuales de clientes en la base de datos del hotel.

Como podemos ver en la figura 5.14 esta ventana consta de un ListView para poder ver la información de los clientes registrados en la base de datos, 4 TextBox para introducir en caso de querer modificar ese campo la información del ID del cliente, Nombre, DNI y Teléfono, un ListBox que permitirá al trabajador seleccionar una nueva habitación para el cliente de una lista con todos los dormitorios. Por último tenemos dos botones, el de Volver que cierra la ventana actual y muestra la ventana superior, en este caso la ventana Clientes, y el botón Actualizar información, que modificará la información de aquellos campos que se hayan rellenado y modificará la disponibilidad de las habitaciones consecuentemente en el caso de que se haya modificado la habitación del cliente. Podemos ver el funcionamiento de este botón representado en la figura 5.29.

- **Habitaciones:**

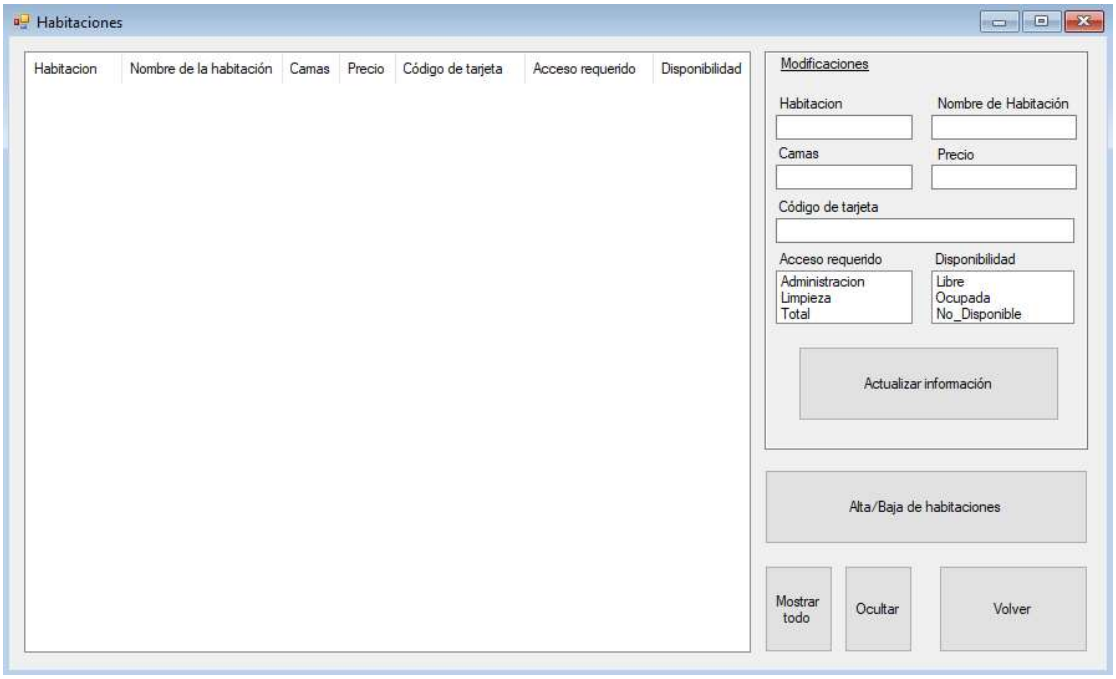


Figura 5.15: ventana Habitaciones

Esta ventana muestra un listado completo de las habitaciones, dividiendo las filas en colores según la disponibilidad de las habitaciones, verde para una habitación libre, rojo para una ocupada y negro para las habitaciones no disponibles para alquilar. Esto facilita al empleado la distinción de las habitaciones rápidamente para saber qué ofrecer a un cliente. También permite modificar cualquier campo de las habitaciones disponibles en la base de datos del hotel.

Como se puede observar en la figura 5.15, la ventana consta de un ListView para mostrar la información de las habitaciones, una sección de modificaciones con 5 TextBox y 2 ListBox con sus correspondientes Labels y el botón Actualizar información, que modificará la información de los campos que se hayan rellenado o seleccionado según el ID de habitación introducido. Este botón funciona siguiendo el esquema de la figura 5.28. También observamos el botón Mostrar todo, que añadirá las estancias de gestión del hotel y uso del personal a la lista cargada inicialmente, en caso de que el empleado lo requiera, el botón Ocultar que oculta la información de las habitaciones mencionadas anteriormente y el botón Volver, que ocultará la ventana actual para mostrar la ventana superior, en este caso Inicio.

- **Alta-Baja de habitaciones:**

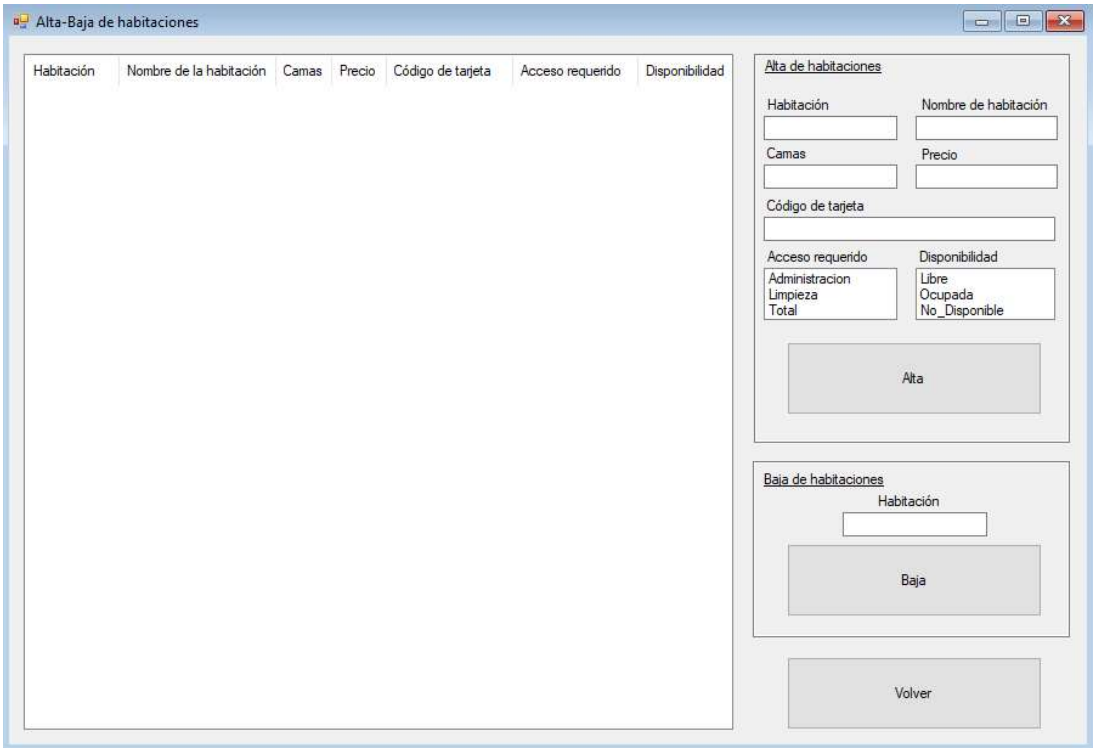


Figura 5.16: Ventana Alta-Baja de habitaciones

En esta ventana el empleado dispondrá de la información completa de las habitaciones, para poder dar de alta una nueva habitación introduciendo las características de la nueva estancia o dar una habitación existente de baja.

Como se observa en la figura 5.16, esta ventana tiene un ListView que muestra la información completa de las habitaciones existentes, una zona de alta de habitaciones con los TextBox y ListBox referenciados con sus respectivos Labels y el botón Alta para dar de alta una nueva habitación en la base de datos del hotel siguiendo el esquema de la figura 5.25, una zona de baja de habitaciones con su TextBox y su correspondiente Label y el botón Baja para eliminar una habitación de las existentes en la base de datos del hotel siguiendo el esquema de la figura 5.26, y por último, el botón Volver que cierra la ventana actual para mostrar la ventana superior, en este caso Habitaciones.

- **Personal:**

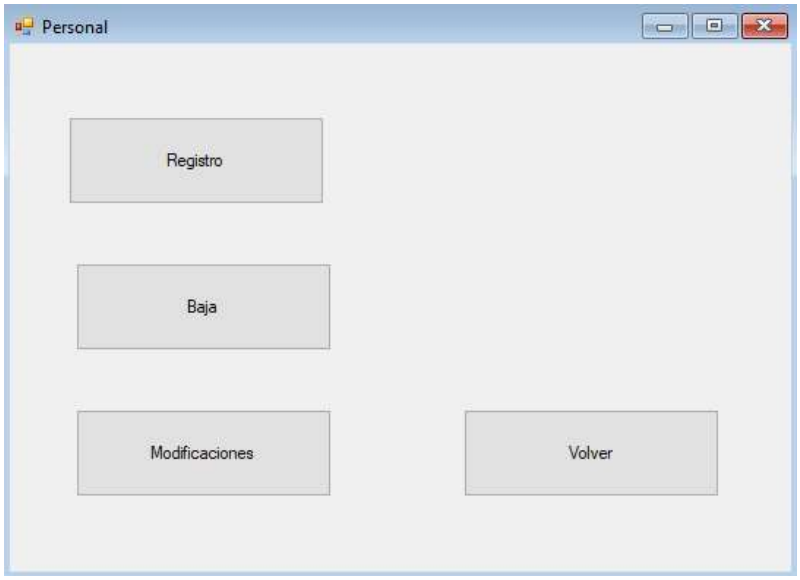


Figura 5.17: Ventana Personal

Esta ventana nos mostrará un menú para gestionar la información de los empleados del hotel o volver a la ventana de inicio.

En la figura 5.17 vemos que la ventana Personal se compone de 4 botones, 3 de ellos encargados de cerrar la ventana actual y abrir cualquiera de las 3 ventanas que gestionan la información de la tabla personal: Registro, Baja y Modificaciones. El cuarto botón es el de Volver, que cierra la ventana actual y abre la ventana superior, en este caso la ventana Inicio.

- **Registro personal:**

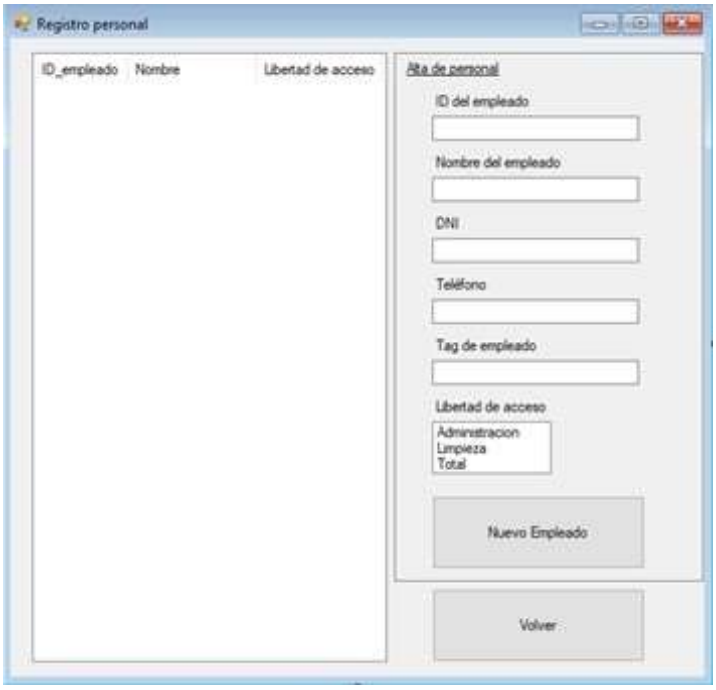


Figura 5.18: Ventana Registro personal

En esta ventana se facilitará la información necesaria para introducir un nuevo empleado en la base de datos del hotel introduciendo los datos necesarios.

Observamos que la figura 5.18 consta de un ListView que mostrará el ID, nombre y libertad de acceso de los empleados actuales en la base de datos del hotel, 4 TextBox y un ListBox con sus respectivos Labels para introducir un nuevo empleado pulsando el botón Nuevo Empleado, que funciona según el esquema de la figura 5.25 y, por último, el botón Volver que cierra la ventana actual y muestra la ventana superior, en este caso la ventana Empleados.

- **Baja personal:**

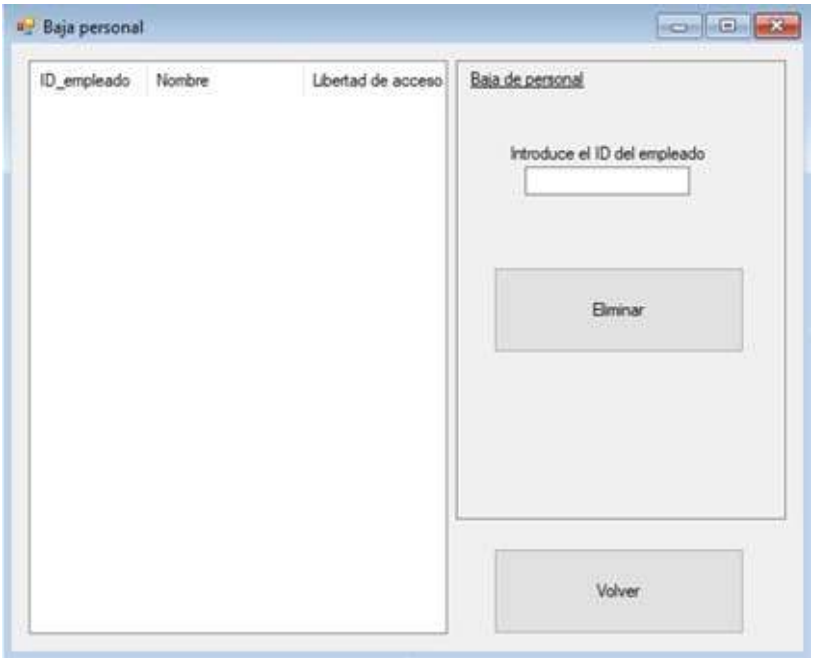


Figura 5.19: Ventana Baja personal

En esta ventana el empleado encargado de manejar la aplicación de escritorio podrá ver la información necesaria para dar de baja a un empleado y llevar a cabo la operación.

Como podemos observar en la figura 5.19 esta ventana tiene un ListView para mostrar la información que deberá introducir en el TextBox que se encuentra acompañado de su respectivo Label. Por último tenemos dos botones, el de Volver que cierra la ventana actual y muestra la ventana superior, en este caso la ventana Personal, y el botón Eliminar, que elimina de la base de datos del hotel al empleado introducido por el empleado si la información ha sido introducida correctamente siguiendo el funcionamiento representado en la figura 5.26.

- **Modificaciones personal:**

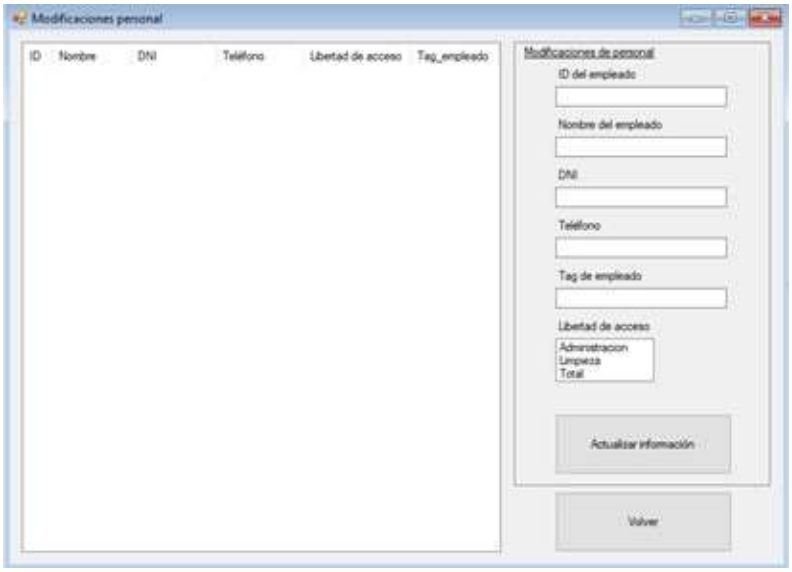


Figura 5.20: Ventana Modificaciones personal

Esta ventana permitirá al empleado modificar la información de cualquier columna de los registros actuales de empleados en la base de datos del hotel.

Como podemos ver en la figura 5.20 esta ventana tiene un ListView para mostrar la información de los empleados que hay actualmente en la base de datos del hotel, 4 TextBox para introducir en caso de querer modificar ese campo la información del ID del empleado, Nombre, DNI y Teléfono, un ListBox para seleccionar la libertad de acceso del trabajador, todos ellos referenciados con sus respectivos Labels. Por último tenemos dos botones, el botón Actualizar información, que modificará la información de aquellos campos que se hayan rellenado de la tabla empleados de la base de datos siguiendo el funcionamiento representado en la figura 5.28, y el botón Volver, que cierra la ventana actual y muestra la ventana superior, en este caso la ventana Personal.

5.2.1 Diseño final de pantallas

El diseño final de pantallas ha sido modificado respecto al planteado inicialmente en el capítulo de análisis y diseño. La parte que ha sufrido más cambios es la del control de las habitaciones. Inicialmente se planteó esta parte con una distribución de pantallas y funciones similar a la de las partes de clientes y empleados, mostrando un menú inicial desde el que seleccionar ver un listado de habitaciones, registrar una nueva habitación o dar de baja una existente.

Sin embargo, dado que las operaciones de alta y baja de habitaciones en la base de datos del hotel son poco comunes, se ha optado por mostrar una primera ventana con el listado de las estancias registradas y una sección para modificar la información de alguna habitación en caso

de necesitarlo, dejando el alta y baja de habitaciones accesible desde un botón en esta primera ventana.

A parte de las habitaciones el diseño de la distribución de pantallas de la aplicación de escritorio no ha cambiado, salvo algunos campos que se ha decidido dar la opción al empleado que manejará la aplicación de seleccionar algunos campos desde listas predefinidas en lugar de tener que introducir la información manualmente en cuadros de texto.

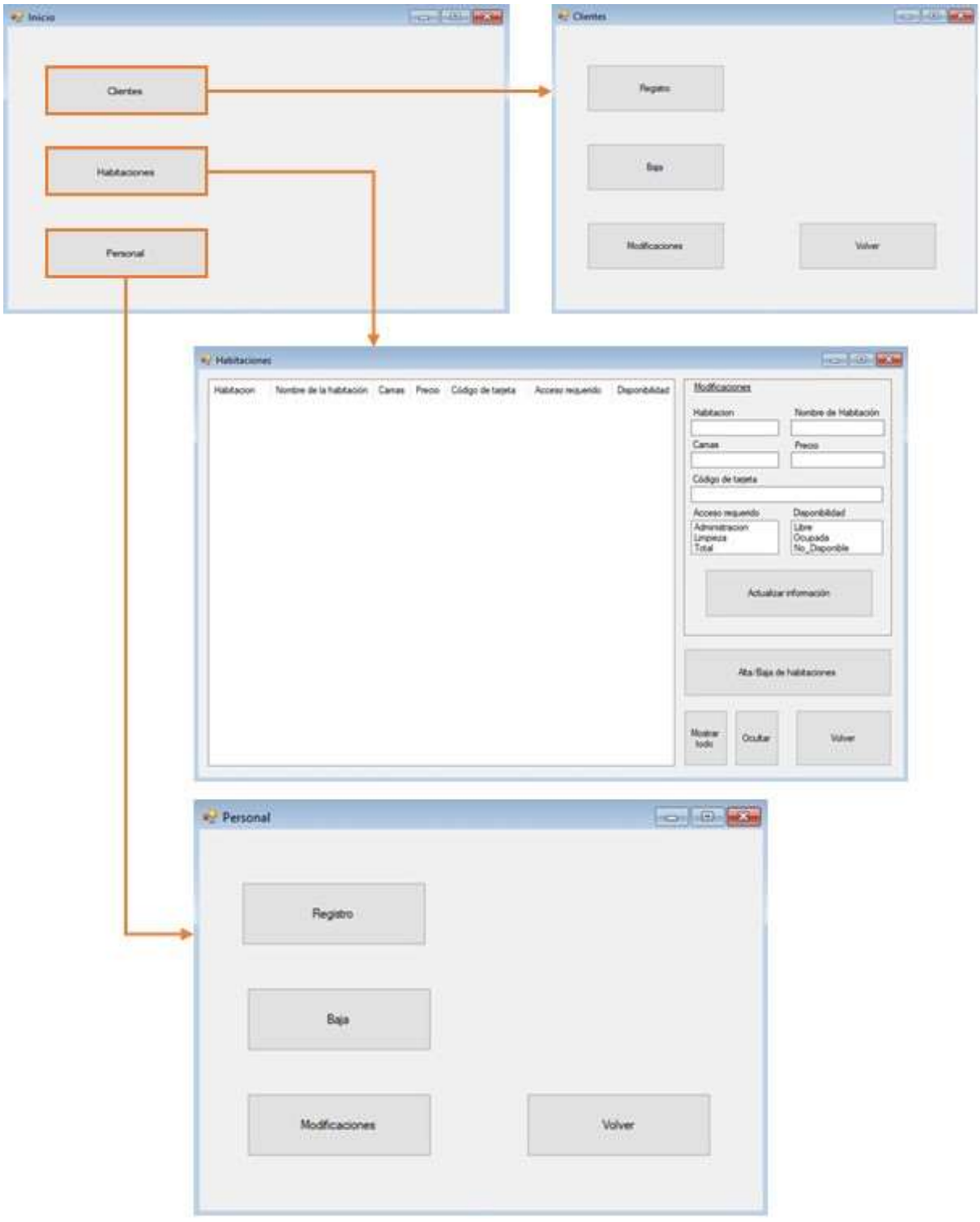


Figura 5.21: Distribución de ventanas desde el Inicio

Figura 5.22: Distribución de ventanas del control de los clientes

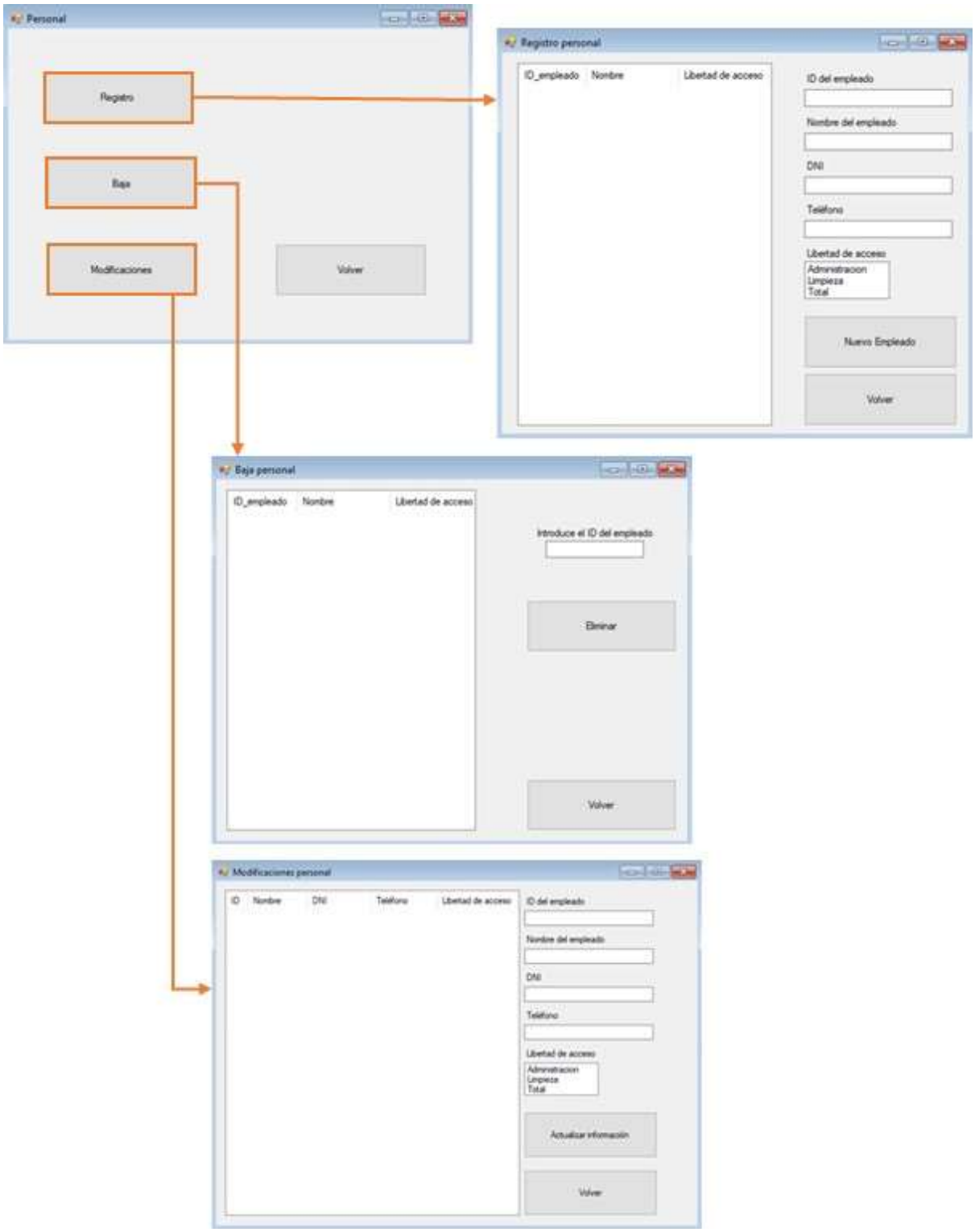


Figura 5.23: Distribución de ventanas del control del personal

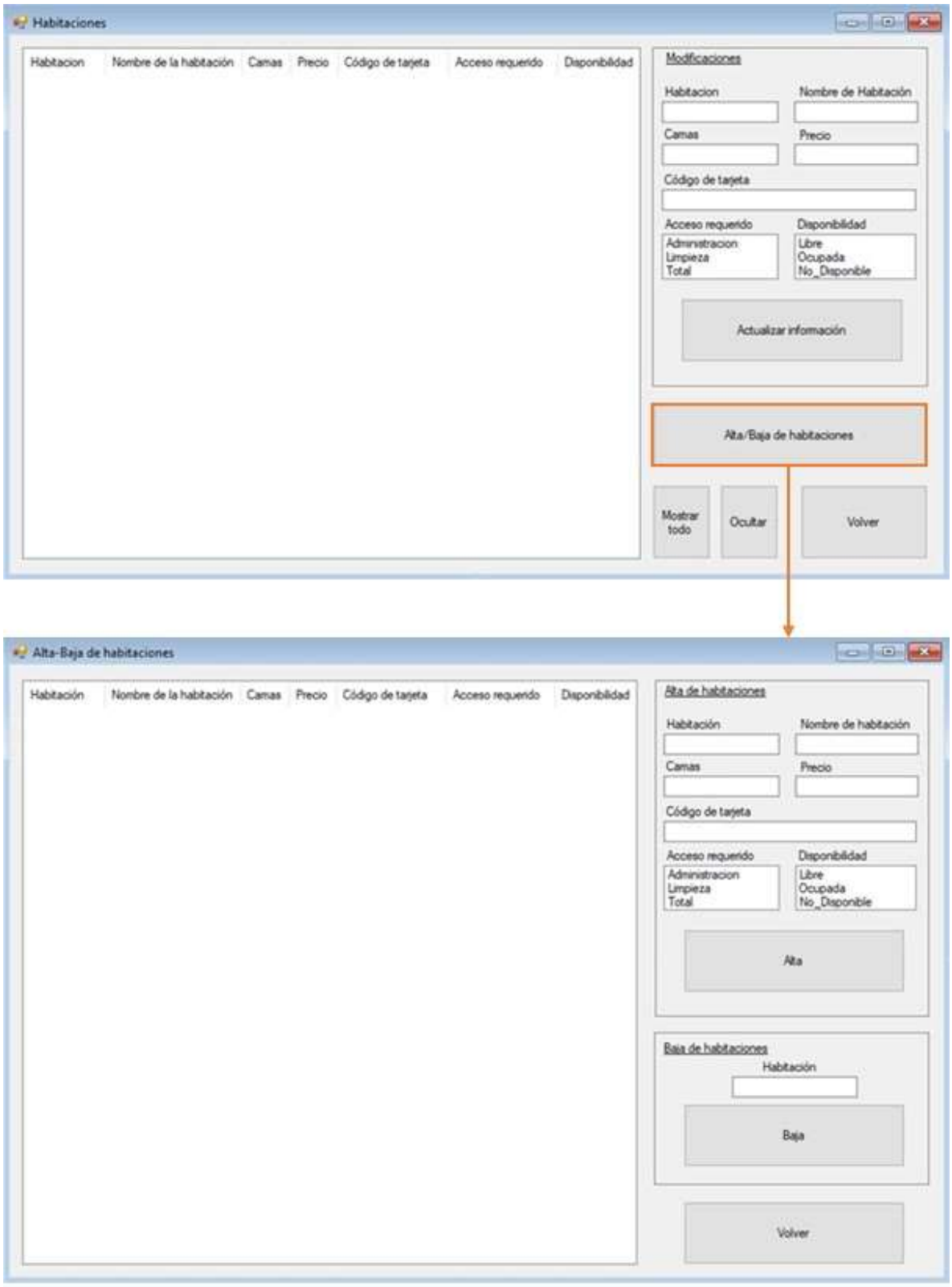


Figura 5.24: Distribución de ventanas del control de habitaciones

5.2.2 Diagramas de flujo de los botones principales

En este apartado se van a mostrar los diagramas de flujo que representan el funcionamiento de los botones principales de la aplicación precedidos de una pequeña explicación de cada uno de ellos.

Primero se muestra el funcionamiento de los botones de registro en el diagrama de la figura 5.25, en este diagrama podemos ver que primero se comprueba que el ID introducido no exista ya, y en caso de ser así se comprueba que se han introducido correctamente el resto de campos del nuevo cliente, empleado o habitación, y en caso de ser así, se llama al PHP encargado de este registro y se le envía la información necesaria para que lo realice. Por último muestra el listado actualizado tras el registro.

A continuación, podemos observar el diagrama de flujo de los botones de baja de empleados y de habitaciones en la figura 5.26 y el de baja de clientes en la 5.27. Estos diagramas funcionan igual salvo que en el botón de baja de clientes también se llamará al PHP encargado de modificar la disponibilidad de la habitación que tenía asignada el cliente a libre. Primero se recogen los IDs de la tabla en cuestión y se comparan con el introducido, si existe se llama al PHP encargado de eliminarlo. Después muestra el listado de la información tras la baja.

Por último, se muestran los diagramas de flujos de los scripts de modificaciones de empleados y habitaciones en la figura 5.28 y modificaciones de clientes en la figura 5.29. Estos diagramas muestran cómo en el código del botón se realiza una comprobación sobre el ID introducido, y en caso de existir se llama al archivo PHP encargado de modificar la información de la tabla correspondiente enviándole los datos a cambiar y cuáles han de ser modificados. En el caso de las modificaciones en clientes, también hace una comprobación sobre si se debe modificar el número de habitación del cliente, y en caso de ser así se cambiará la antigua a libre y la nueva a ocupada antes y después de llamar al PHP para modificar información respectivamente. Por último se mostrará la información actualizada tras la modificación.

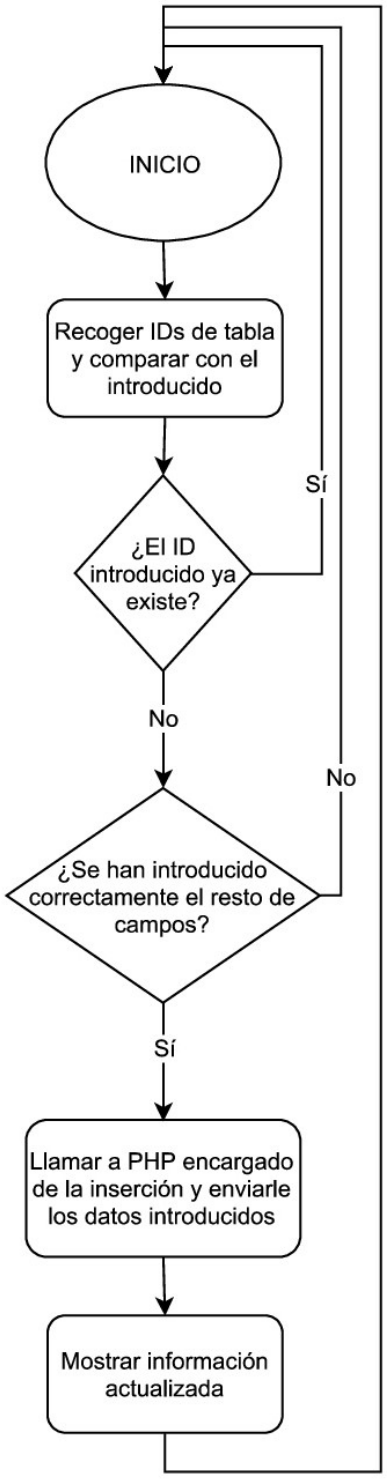


Figura 5.25: Registro de clientes, empleados y habitaciones.

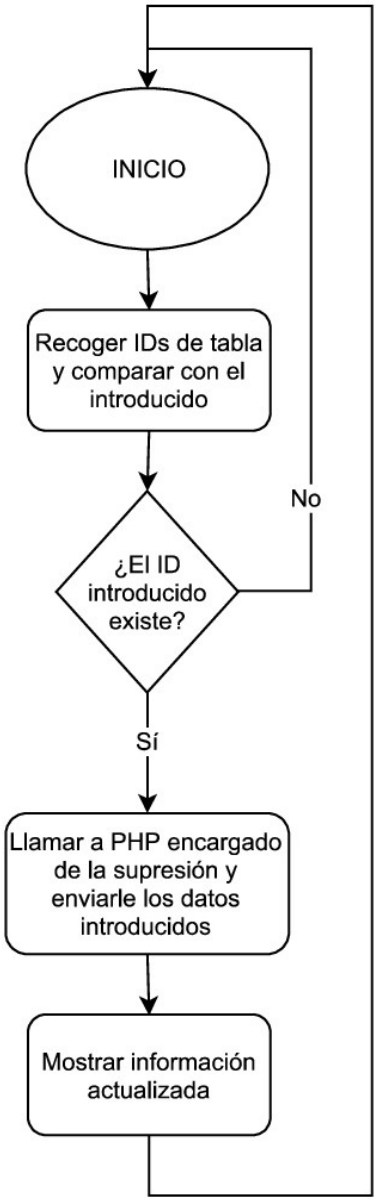


Figura 5.26: Baja de empleados y habitaciones.

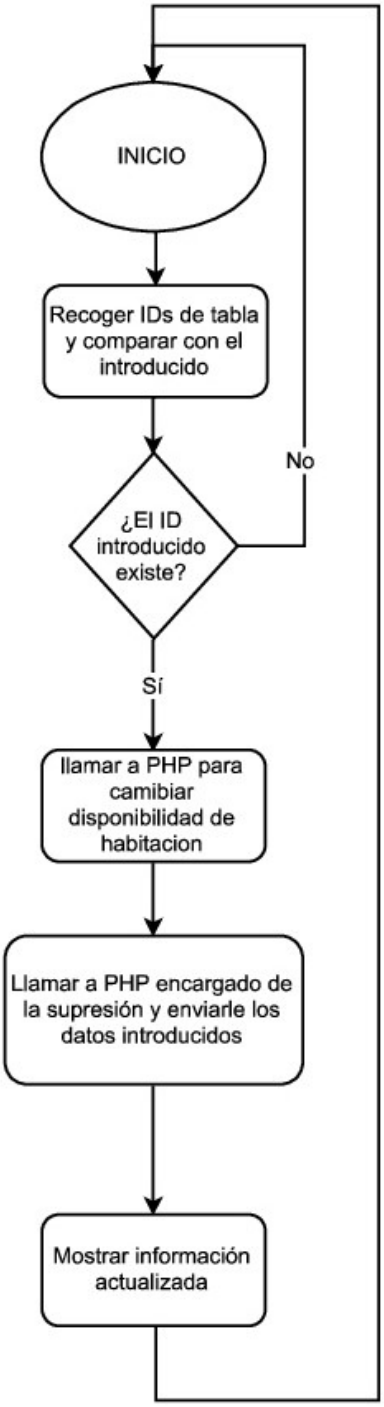


Figura 5.27: Baja de clientes

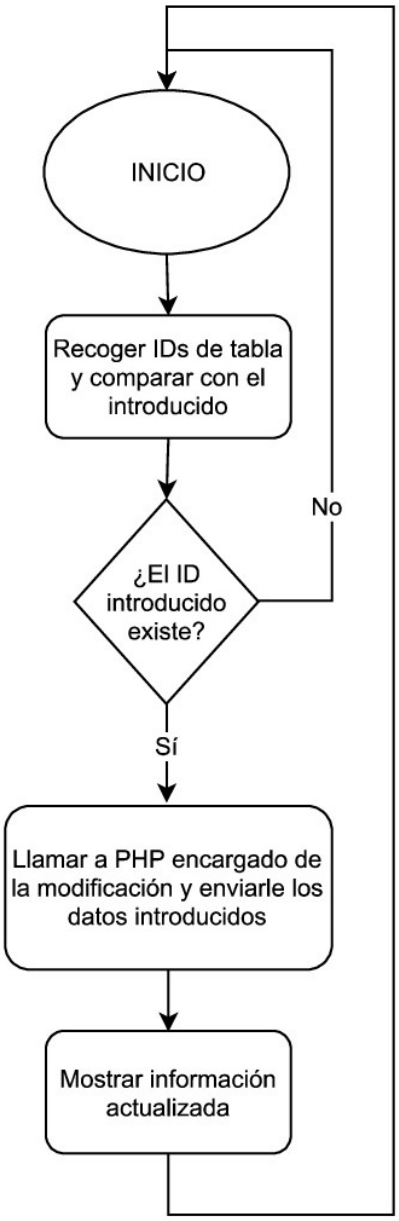


Figura 5.28: Modificaciones de empleados y habitaciones

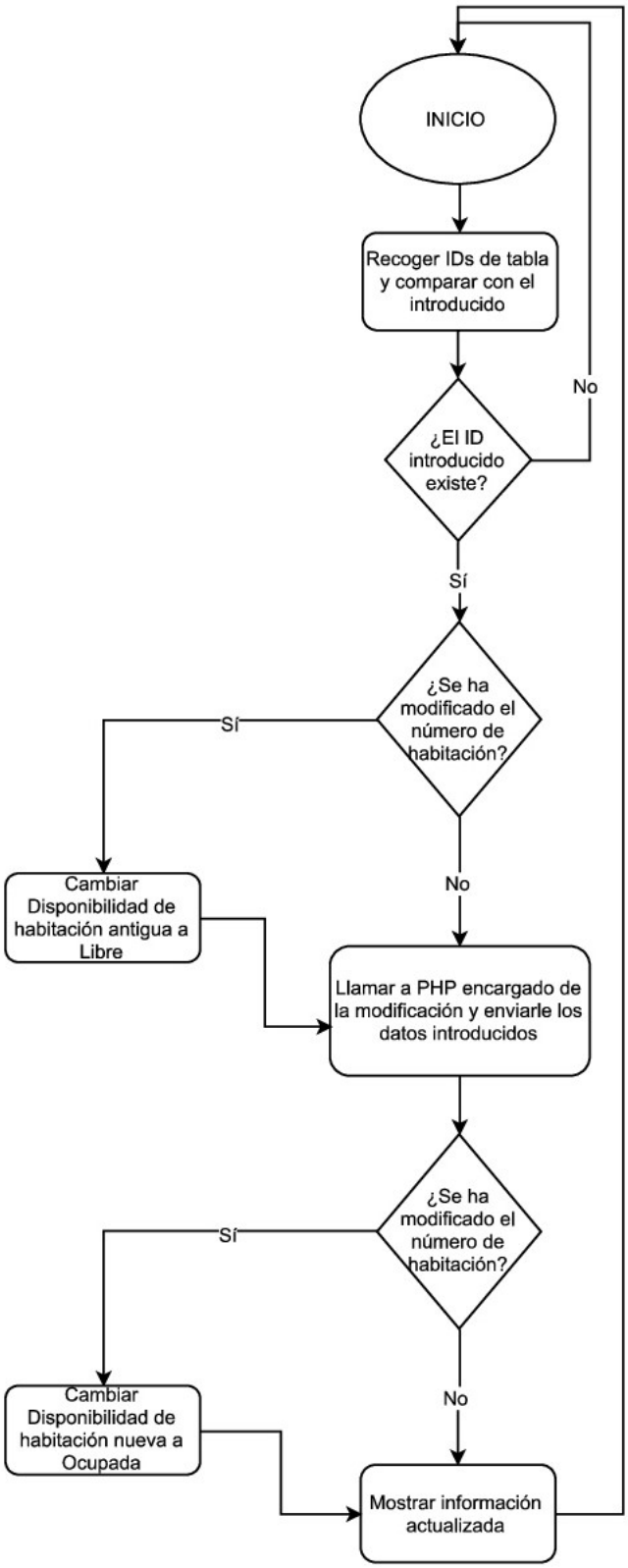


Figura 5.29: Modificaciones de clientes

5.3 Desarrollo de scripts PHP

En el desarrollo de los scripts PHP podemos dividir estos en dos grupos, los scripts a los que se accede desde la aplicación de control en Python y a los que se accede desde la aplicación de escritorio, que, a su vez, podemos dividir en scripts de listado, registro, baja y modificaciones.

Para todo esto se han tenido que solucionar los problemas que se detallan del punto 5.3.1.1 al 5.3.1.5, mostrando de cada uno de ellos una breve descripción, la solución implementada y un ejemplo de la misma.

Por último, se pueden observar los diagramas de flujo que representan el funcionamiento de cada uno de los scripts PHP utilizados en el punto 5.3.2.

5.3.1 Problemas abordados

5.3.1.1 Conexión con base de datos

La función de todos los scripts PHP utilizados en el sistema es servir de enlace entre los clientes y la base de datos alojada en el servidor, por lo que este es el primer problema a tratar. Para esto utilizaremos la función mysqli_connect(), con la cual podremos almacenar la respuesta ante un intento por conectar con la base de datos en una variable de nuestro script. Para conectar con la base de datos deberemos de introducir por este orden el nombre del host, nombre de usuario, contraseña y base de datos a la que queramos acceder como parámetros de la función mysqli_connect. En la figura 5.30 podemos ver un ejemplo en el que se guardaría la respuesta ante este intento en la variable "$link" conectándose con el host local o "localhost", mediante el nombre de usuario "root" con contraseña "electronica" y accediendo a la base de datos con el nombre "sql11191051". A continuación hace una comprobación de que la conexión se ha realizado con éxito y en caso contrario devuelve un mensaje informando del error.

```
// Conexión con la base de datos
$link = mysqli_connect("localhost", "root", "electronica", "sql11191051");

// Comprobamos que se ha conectado correctamente
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
```

Figura 5.30: conectar con una BD desde PHP

5.3.1.2 Ejecutar PHP recibiendo parámetros

El funcionamiento de la mayoría de scripts PHP utilizados requiere ejecutar un código que interaccione con la base de datos en función de una o más variables enviadas desde cualquiera de los clientes del sistema. Para esto debemos declarar una variable comenzando con el símbolo "$" e igualarla al valor recibido, en nuestro caso, por POST, con el nombre con el que se ha enviado dicha variable desde el proceso que ha llamado a nuestro script. A continuación, podemos ver en la figura 5.31 un ejemplo de una variable recibida por POST que deberá ser enviada desde el proceso que llame al script con el nombre "ID_Empleado" y se guardará en

nuestro script con el mismo nombre, siendo este modificable cambiando el nombre de la variable que empieza por "$".

```
$ID_Empleado = $_POST['ID_Empleado'];
```

Figura 5.31: crear variable en PHP recibiendo parámetro por POST

5.3.1.3 Ejecutar una sentencia sobre la base de datos

Una vez que sabemos cómo conectar con una base de datos y cómo recibir parámetros desde procesos externos, debemos declarar y lanzar una sentencia SQL sobre la base de datos. Para esto primero declararemos la sentencia igualándola a una variable de PHP, y después utilizaremos la función mysqli_query() para ejecutar dicha sentencia sobre nuestra base de datos enviándole como parámetros a la función, por este orden, la variable en la que guardamos la respuesta de conexión con la base de datos y la variable en la que hemos definido nuestra sentencia. Podemos ver un ejemplo de uso de la función mysqli_query() en la figura 5.32, en cuyo caso declararemos una sentencia "DELETE" y la enviaremos utilizando como variable de conexión con la base de datos la del ejemplo visto en el punto 5.3.1.1.

```
// Declaramos la sentencia SQL y la lanzamos a la BD
$query  = "DELETE FROM empleados WHERE ID_Empleado = '$ID_Empleado' ";
$result = mysqli_query($link,$query);
```

Figura 5.32: declaración y ejecución de sentencia SQL sobre BD desde PHP

5.3.1.4 Devolver información a otros procesos

Para poder realizar una correcta conexión entre los clientes y la base de datos mediante los scripts PHP, deberemos de ser capaces no solo de recibir información sino también de mandarla. Esto lo haremos de dos formas distintas, en el caso de la conexión con la aplicación de escritorio devolveremos una respuesta con formato XML que desde el programa en C# traduciremos para poder recorrerlo y utilizar la información que nos interese. Podemos ver un ejemplo de cómo se devuelve la información del script utilizado para mostrar un listado de clientes en la aplicación de escritorio en la figura 5.33, en la que las líneas de código del bucle while serían una sola, pero se han dividido para facilitar su representación en este documento. En el caso del programa de control en Python, los scripts PHP crearán un fichero .xml externo al que se accederá después de llamar al script desde el programa en Python para traducirlo y utilizar la información que contenga. Esto se ha implementado así debido a la falta de librerías en Python que generen una variable que contenga la respuesta de llamar a una dirección web y aplicar una decodificación mediante la función parse(), por lo que primero se llama a la dirección web y a continuación se traduce la información aplicando la función parse() al archivo generado. En la figura 5.34 se puede observar el procedimiento para generar un archivo .xml con la información resultante de una llamada a la base de datos. Primero utilizamos la función mysqli_fetch_assoc() para obtener una fila de resultado como un array asociativo y poder enviar de este modo las distintas variables agregando este array y su tag identificativo a continuación, como podemos ver en la parte en la que se crea la variable $acceso. A continuación crearemos los distintos elementos de nuestro archivo .xml y por último lo guardaremos especificando una ruta y un nombre de archivo.

```
// Devolvemos la información obtenida por la consulta en formato XML
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
echo "<MessageXML>";
while($row = mysqli_fetch_array($result))
{
  echo "<Data>"."<ID_Cliente>{$row['ID_Cliente']}</ID_Cliente> "."<Nombre>{$row['Nombre']}</Nombre>"
  ."<Habitacion>{$row['Habitacion']}</Habitacion>"."</Data>";
}
echo "</MessageXML>";
```

Figura 5.33: devolver información en formato XML desde PHP

```
// Creamos y guardamos el XML con la información devuelta por la BD
        $row = mysqli_fetch_assoc($result);

        $xml = new DomDocument('1.0', 'UTF-8');

        $MsgXML = $xml->createElement('MsgXML');
        $MsgXML = $xml->appendChild($MsgXML);

        $Data = $xml->createElement('Data');
        $Data = $MsgXML->appendChild($Data);

        $acceso = $xml->createElement('Acceso_requerido', $row['Acceso_requerido']);
        $acceso = $Data->appendChild($acceso);

        $xml->formatOutput = true;
        $el_xml = $xml->saveXML();
        $xml->save('C:/xampp/htdocs/TFG/Habitacion-Libertad.xml');
```

Figura 5.34: crear un archivo XML desde PHP con información obtenida de la BD

5.3.1.5 *Cerrar la conexión con la base de datos*

Por último, deberemos de cerrar la conexión creada con la base de datos antes de finalizar nuestro script. Para esto utilizaremos la función mysqli_close() a la que le pasaremos como parámetro la variable en la que hemos guardado la respuesta del intento de conexión con la base de datos. Podemos ver un ejemplo en la figura 5.35, a la que se le manda como parámetro la variable en la que se guardó la respuesta de conexión con la base de datos en el ejemplo del punto 5.3.1.1.

```
// Cerramos conexión
mysqli_close($link);
```

Figura 5.35: cerrar conexión con la BD

5.3.2 Diagramas de flujo de los scripts PHP

En este apartado se incluirán diagramas de flujo de cada script PHP utilizado en el proyecto a fin de clarificar el funcionamiento de los mismos. Primero se mostrarán los scripts de listado, representados en la figura 5.36, puesto que todos ellos actúan del mismo modo, conectándose con la base de datos, posteriormente declarando y ejecutando una consulta SELECT con la información que se desea listar y se envía la información obtenida en formato XML. La

diferencia entre los 3 scripts consiste, en la declaración de la sentencia SELECT, que se cargará sobre una de las 3 tablas en función de las necesidades, y en el envío de esta información con sus respectivas etiquetas XML.

A continuación tenemos los diagramas referentes a los scripts de alta, primero tenemos el de baja de clientes en la figura 5.37 que funciona de forma similar a los de alta de empleados y de habitaciones mostrados en la figura 5.38, con la diferencia de que este script también se encarga de ejecutar una sentencia UPDATE con la que cambiar la disponibilidad de la habitación que se le asigna al nuevo cliente a ocupada. Salvando esa diferencia ambos diagramas muestran como se conecta con la base de datos, declaran las variables recibidas por POST y se ejecuta la sentencia INSERT en la tabla pertinente.

Después se mostrarán los diagramas de flujo referentes a los scripts de baja en las figuras 5.39 el de baja de clientes y 5.40 el de baja de empleados y habitaciones, cuyo funcionamiento es análogo al de los scripts de alta, solo que la sentencia a ejecutar en este caso será una sentencia DELETE, y en el caso del script de baja de clientes también existe una pequeña diferencia respecto a los de baja de empleados y habitaciones, en este caso la disponibilidad de la habitación que tenía asignada el empleado pasa a ser libre.

Posteriormente se muestran los diagramas de flujos que muestran el funcionamiento de los scripts de modificación de información, en este caso hay un diagrama para cada script, pudiéndose observar los de modificaciones de clientes, empleados y habitaciones en las figuras 5.41, 5.42 y 5.43 respectivamente. Estos scripts consisten en la misma conexión con la base de datos y declaración de variables que los anteriores, solo que se irá comprobando si hay que cambiar cada atributo de la tabla sobre la que se esté trabajando y se cargará una sentencia UPDATE en caso de que se deba cambiar dicho atributo. Esta comprobación se realiza mediante variables enviadas por POST.

Por último, podemos observar los diagramas de flujo que representan el funcionamiento de los scripts destinados al control de acceso en las habitaciones. Estos scripts se pueden representar todos mediante el mismo diagrama, puesto que funcionan de la misma forma solo que cada uno pide una información distinta. De igual forma que los scripts de listado se conectan con la base de datos, declaran las variables recibidas por POST y ejecutan una sentencia SELECT sobre la base de datos para obtener la información requerida, solo que en lugar de devolver la información directamente en formato XML en este caso se genera un archivo con formato XML con dicha información para poder leerlo posteriormente desde el programa desarrollado en Python. Por último, en todos los scripts se cierra la conexión establecida con la base de datos.

5.3.2.1 Scripts de listado

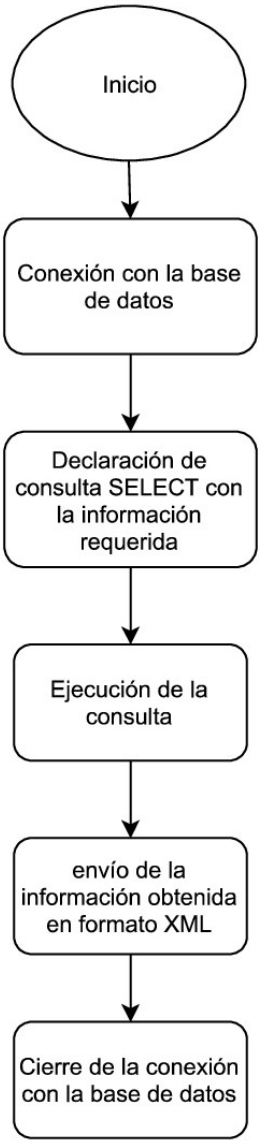


Figura 5.36: Funcionamiento de los scripts PHP Lista_clientes, Lista_empleados y Lista_habitaciones

5.3.2.2 Scripts de alta

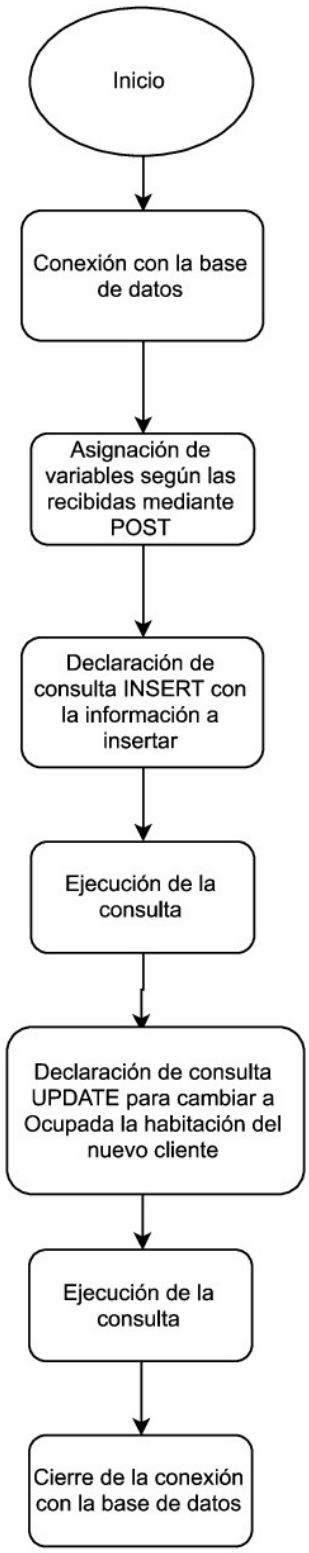


Figura 5.37: Funcionamiento del script PHP Nuevo_cliente

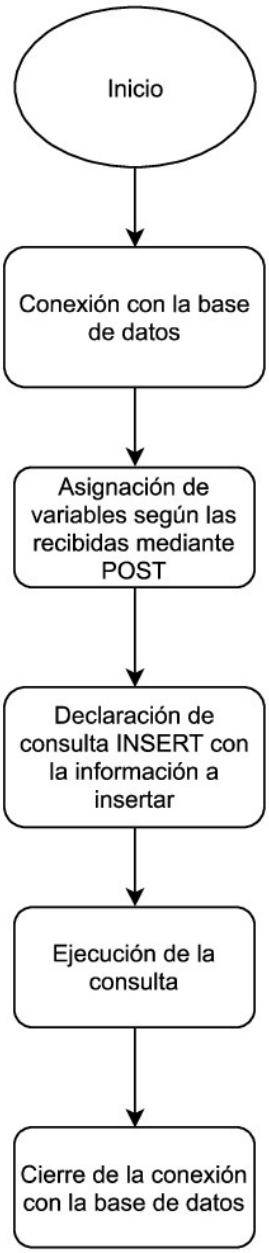


Figura 5.38: Funcionamiento del script PHP Nuevo_empleado y Nueva_habitacion

5.3.2.3 Scripts de baja

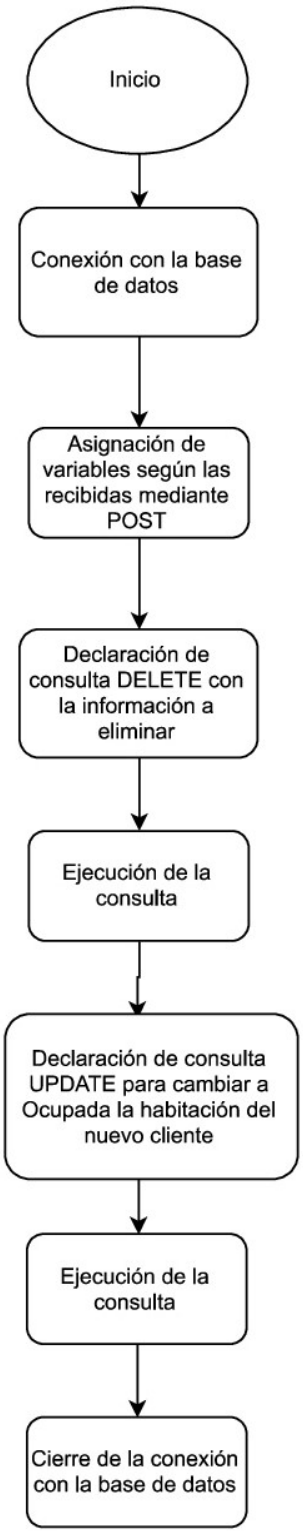


Figura 5.39: Funcionamiento del script PHP Baja_clientes

Figura 5.40: Funcionamiento del script PHP Baja_empleados y Baja_habitaciones

5.3.2.4 Scripts de modificaciones

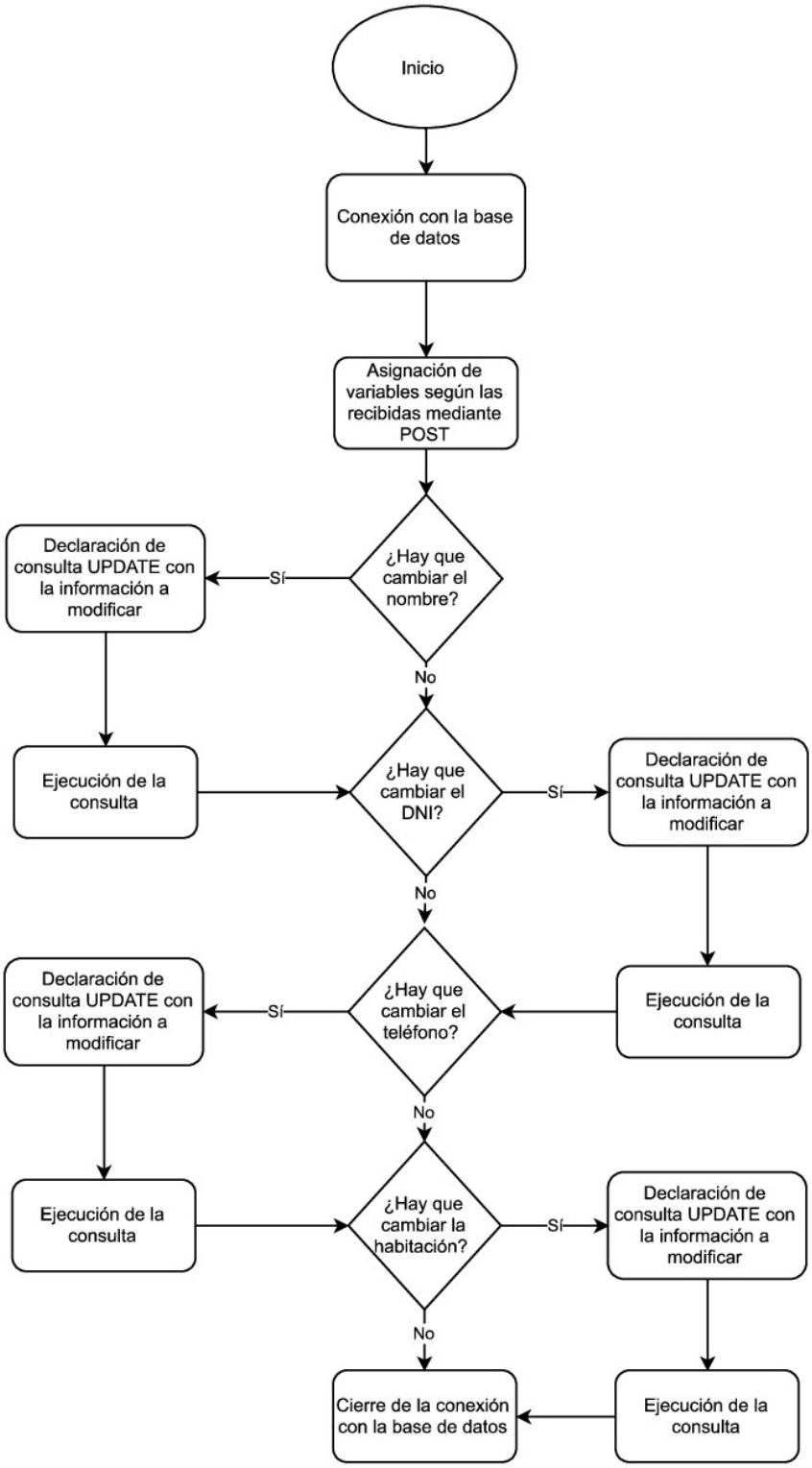


Figura 5.41: Funcionamiento del script PHP Modificaciones_clientes

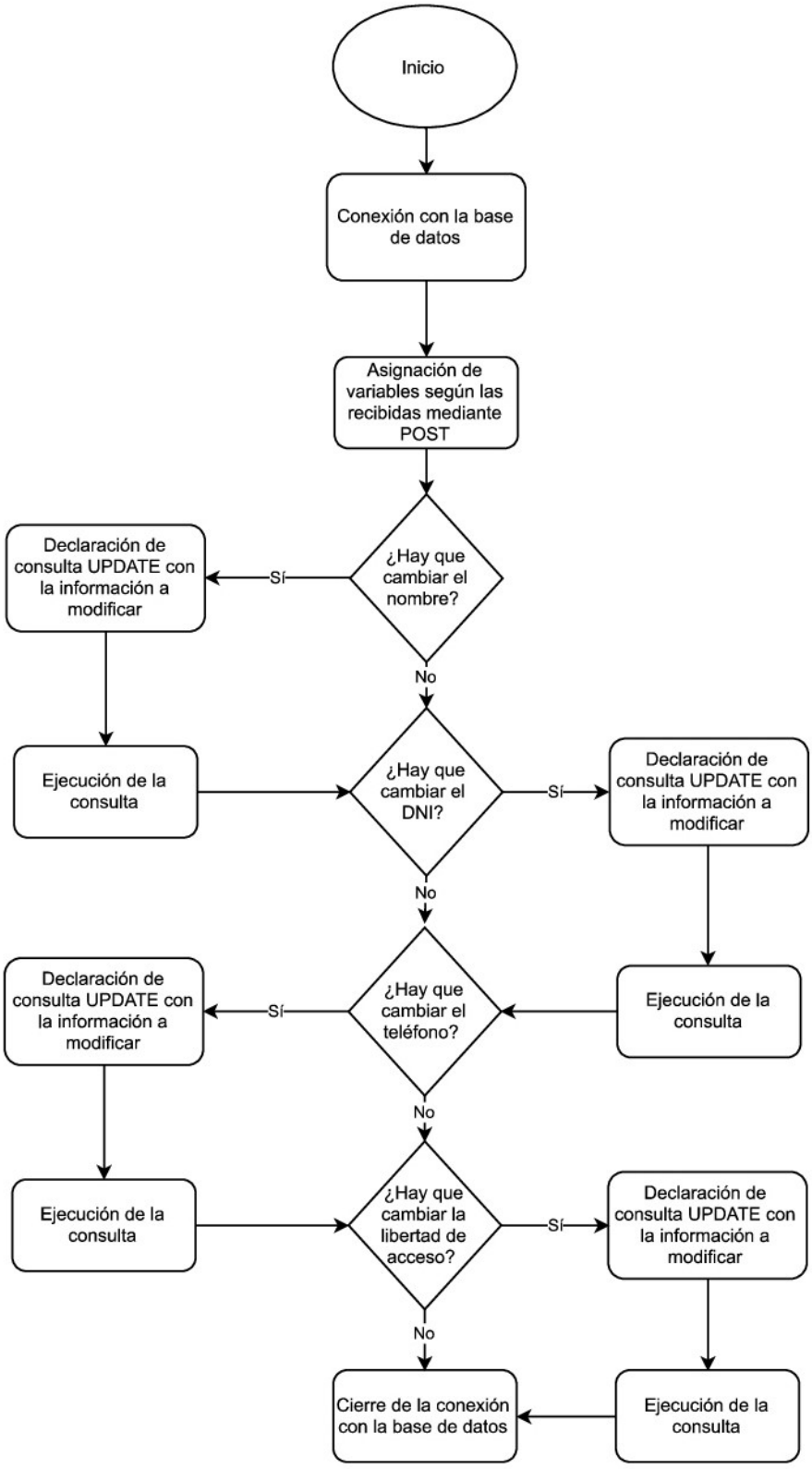


Figura 5.42: Funcionamiento del script PHP Modificaciones_empleados

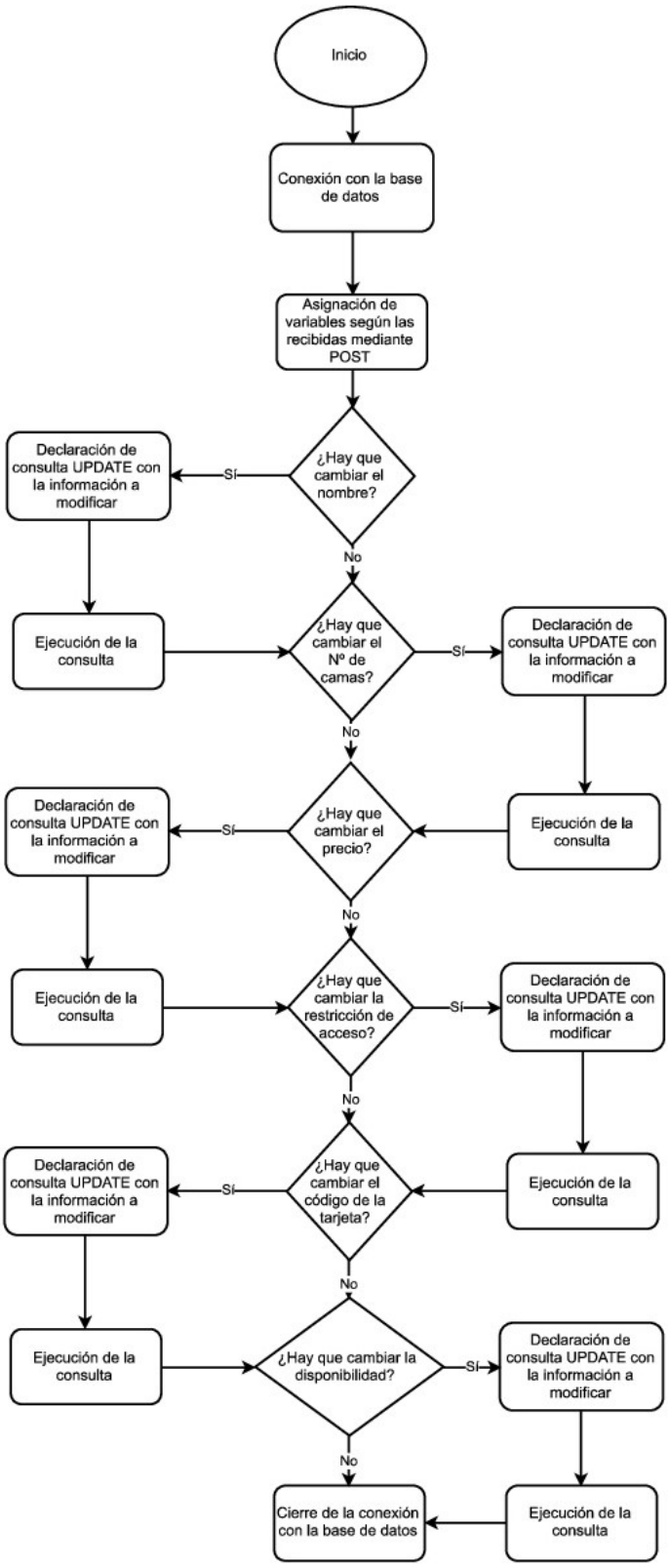


Figura 5.43: Funcionamiento del script PHP Modificaciones_habitaciones

5.3.2.5 Scripts para control de acceso

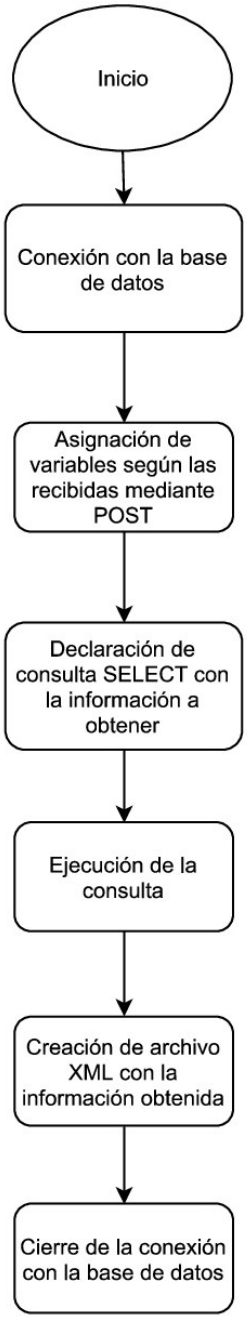


Figura 5.44: Funcionamiento de los scripts ID-Libertad_Acceso, Habitacion-Tarjeta y Habitacion-Libertad_Acceso

5.4 Configuración de Raspbian y ajustes necesarios

Para poder utilizar todos los elementos necesarios de la parte de control de acceso, deberemos instalar los paquetes necesarios en nuestro sistema operativo Raspbian y modificar la configuración de nuestra Raspberry Pi para que permita utilizar todos los elementos conectados a esta.

Primero habilitaremos la comunicación SPI de la Raspberry. La interfaz SPI se encuentra deshabilitada por defecto, para habilitarla ejecutamos el siguiente comando en la terminal:

sudo raspi-config

Una vez en el menú de raspi-config, habilitaremos la opción SPI desde Advanced options. Por último, reiniciamos la Raspberry para aplicar los cambios realizados y que el módulo SPI quede activado.

A continuación, realizaremos un update para obtener las versiones más recientes de los paquetes que vamos a instalar. Para esto introduciremos el siguiente comando en la terminal:

sudo apt-get update

Para realizar una actualización de los paquetes ya instalados ejecutaremos un upgrade del sistema introduciendo el siguiente comando en la terminal:

sudo apt-get upgrade

Posteriormente ejecutaremos el siguiente comando para instalar Python:

sudo apt-get install python-dev

Sin embargo, no todos los paquetes de Python están disponibles en los repositorios de Raspbian, incluso algunos pueden estar desactualizados. Por este motivo, vamos a instalar la herramienta PIP mediante el siguiente comando:

sudo apt-get install python-pip

Seguidamente vamos a instalar el controlador para SPI en nuestra Raspberry, dado que el lector RFID utiliza el bus SPI para conectarse con nuestra placa. Para esto primero vamos a instalar la herramienta Git para clonar e instalar el controlador mediante el siguiente comando:

sudo apt-get install git

Ahora clonamos el código fuente del controlador y lo instalamos mediante los siguientes comandos:

git clone https://github.com/lthiery/SPI-Py.git

cd SPI-Py

sudo python setup.py install

Después instalaremos los paquetes necesarios para el funcionamiento del display LCD y para poder manejar imágenes con él, esto lo haremos introduciendo los siguientes comandos:

sudo apt-get install build-essential python-dev

sudo apt-get install python-smbus python-pip

sudo apt-get install python-imaging python-numpy

sudo pip install RPi.GPIO

Por último, clonaremos e instalaremos el controlador Adafruit para ILI9341 para manejar el display LCD introduciendo los siguientes comandos en la terminal:

git clone https://github.com/adafruit/Adafruit_Python_ILI9341.git

cd Adafruit_Python_ILI9341

sudo python setup.py install

Con todos los paquetes instalados y la comunicación SPI habilitada, introduciremos los siguientes 3 comandos en la terminal para enviar un '1' a la salida del GPIO 18 de la Raspberry, maximizando así la luminosidad del display LCD:

sudo echo 18 > /sys/class/gpio/export

sudo echo out > /sys/class/gpio/gpio18/direction

sudo echo 1 > /sys/class/gpio/gpio18/value

5.5 Desarrollo de programa para control de acceso

En el desarrollo del programa para control de acceso a las puertas del hotel, se ha partido de un software proporcionado por la empresa suministradora del lector RFID, la Hat-Board y el display LCD, IngeniApp. Esta empresa facilita el software, así como los archivos de documentación asociados al mismo y permite su uso sin limitaciones, incluyendo, sin límites, los derechos de uso, copia, modificación, combinación, publicación, distribución, sublicencia y / o venta de copias del Software a todo el que obtenga una copia del mismo a través de su página web sujeto a las siguientes condiciones:

- El aviso de copyright anterior y el siguiente aviso de permiso se incluirán en todas las copias o partes sustanciales del software.
- El software se proporciona "tal como está", sin garantía de ningún tipo, explícita o implícita, incluyendo pero no limitando a las garantías de comerciabilidad y adecuación para un fin determinado. En ningún caso, los autores o los titulares de derechos de autor serán responsables de cualquier reclamación, daño u otra responsabilidad, ya sea en una acción de contrato, agravio o de otra manera, que surja de o en relación con el software, uso u otros negocios relacionados con el software.

Este software consiste en un programa que después de inicializar el estado de los componentes que se van a utilizar, se mantiene esperando en un bucle while llamando cada 0.1 segundos a la función encargada de detectar si se ha acercado una tarjeta al lector. Cuando detecta una tarjeta muestra un mensaje estándar por pantalla saludando al usuario.

En la modificación del software se ha cambiado el modo en que este actúa cuando se detecta una tarjeta y las notificaciones o mensajes mostrados mediante el LCD, dado que, aunque el programa se ejecutará desde la terminar de la Raspberry la única forma que tendrá el usuario

de obtener información del programa será mediante el display, puesto que no habrá una pantalla conectada a la Raspberry a modo de pantalla.

Estas modificaciones se han realizado principalmente en la función Mifare(), que es la encargada de la detección de tarjetas y las acciones a llevar a cabo cuando una tarjeta es detectada. El funcionamiento del programa en general y la función Mifare() en particular podemos verlos en las figuras 5.50 y 5.51 respectivamente.

5.5.1 Problemas abordados

A continuación, se detallarán los problemas a tratar dentro del propio código en Python del programa, así como la solución a la que se ha llegado y el fragmento de código con el que se ha resuelto.

5.5.1.1 Distinción entre tarjetas de cliente y de empleado

Para la distinción entre los dos tipos de tarjetas, se ha partido de la identificación que poseían tanto la tarjeta blanca como el tag suministrados junto con el lector RFID, la cual está dividida en 4 segmentos y de la que se pueden distinguir entre empleados y clientes valorando la longitud del primer segmento, expresado en el programa como "uid[0]". Aquel identificador cuyo uid[0] tenga una longitud de 2 dígitos será un tag de empleado, mientras que el que tenga una longitud de 3 dígitos será una tarjeta de cliente. En la figura 5.45 podemos ver las sentencias if con una función len() en su comparación que nos permitirán distinguir entre empleados y clientes.

```
if len(str(uid[0])) == 2:


if len(str(uid[0])) == 3:
```

Figura 5.45: Distinción entre tarjeta de cliente y de empleado

5.5.1.2 Llamar a un archivo PHP enviándole parámetros mediante POST

El siguiente problema en el desarrollo del control de acceso es la llamada a scripts PHP enviándoles parámetros mediante POST. Esto se resuelve mediante el uso de las funciones urlencode() y ulropen(), pertenecientes a la librería urllib. La función urlencode() se utilizará para asignar un nombre y valor a una o más variables para posteriormente enviarlas como parámetros en la llamada al script. La función urlopen() se encarga de realizar la llamada al PHP añadiendo como parámetros los incluidos en la variable generada mediante urlencode().

En la figura 5.46 se puede ver un ejemplo de este proceso en el que se ha generado una variable "campos" con el campo 'Tag_empleado' y el valor 'ID_empleado' que se corresponde con el ID de la tarjeta leída por el lector en caso de que esta sea un tag de empleado. A continuación se llamará al script encargado de obtener la libertad de acceso de un empleado en función de su tag de empleado enviándole como parámetro la variable "campos".

```
ID_empleado = str(uid[0]) + str(uid[1]) + str(uid[2]) + str(uid[3])

campos = urllib.urlencode({"Tag_empleado":ID_empleado})
sitio = urllib.urlopen("http://192.168.1.105:80/TFG/Tag-Libertad_acceso.php",campos)
```

Figura 5.46: llamada a PHP enviando parámetros mediante POST

5.5.1.3 *Abrir un archivo XML y obtener la información que contiene*

Una vez sabemos cómo llamar al PHP que generará un archivo XML con la información requerida, necesitamos ser capaces de abrir remotamente dicho archivo XML y obtener la información que contiene para utilizarla en el programa. Para esto primero abriremos el archivo con la función parse() de la librería untangle, tras lo cual podremos acceder a los distintos elementos que contiene el XML tal y como se ve en la figura 5.48, en la que se recogerán los datos de las variables Nombre y Libertad_acceso contenidas en los nodos Data y MsgXML. En la figura 5.47 podemos ver el estado del archivo XML al que accederemos a fin de clarificar la estructura del mismo y como se accede a sus datos.

```
<?xml version="1.0" encoding="UTF-8"?>
- <MsgXML>
  - <Data>
      <Libertad_Acceso>Administracion</Libertad_Acceso>
      <Nombre>Alberto Sanz</Nombre>
    </Data>
</MsgXML>
```

Figura 5.47: Archivo XML

```
url = 'http://192.168.1.105:80/TFG/Tag-Libertad.xml'
parsed_data = untangle.parse(url)
Libertad_acceso = parsed_data.MsgXML.Data.Libertad_Acceso
Nombre = parsed_data.MsgXML.Data.Nombre
```

Figura 5.48: Método para abrir un archivo XML y obtener el valor de sus variables

5.5.1.4 *Mostrar información mediante el display LCD*

Para abordar la información que se mostrará mediante el display LCD dividiremos este problema en dos partes, por un lado el estado “inactivo” del programa, en el que tan solo se estará escuchando a la espera de detectar una tarjeta y el estado “activo” del programa, aquello que se realizará cuando se detecte una tarjeta.

En lo referente al estado “inactivo” el display mostrará un mensaje en la parte superior informándonos de que podemos acercar una tarjeta al lector y en la parte inferior el día y la hora actuales. Esto se especificará en la función en la que se inicializa el estado del display, debido a que cuando se detecte una tarjeta, el display mostrará otra información durante unos segundos para volver a mostrar el mensaje y la hora especificados en la inicialización posteriormente. En la figura 5.49 podemos ver la forma de establecer el mensaje y posteriormente la llamada a la función show_time(), la cual se observa en la figura 5.50 y que muestra un cuadro con el día y la hora actuales.

```
init()
# Write two lines of white text on the buffer, rotated 90 degrees counter clockwise.
# draw_rotated_text(disp.buffer, 'Hello World!', (150, 100), 90, font, fill=(255,255,255))
draw_rotated_text(disp.buffer, 'Acerque una', (180, 50), 270, font, fill=(107, 142, 35))
draw_rotated_text(disp.buffer, 'tarjeta', (135, 90), 270, font, fill=(107, 142, 35))

# Write buffer to display hardware, must be called to make things visible on the display!
image = Image.open('azul.jpg')
disp.display(image)
pasos = 0
# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()
show_time()
```

Figura 5.49: Inicialización del estado del display

```
def show_time():
    logging.info("Starting")
    cadtime = " "
    cadtime = time.strftime("  %d/%m/%y  ") + time.strftime("%H:%M")
    draw_rotated_text(disp.buffer, cadtime, (30, 25), 270, font, fill=(107, 142, 35))
    disp.display()
```

Figura 5.50: Definición de función show_time()

5.5.1.5 *Generar un registro del acceso a la habitación en un documento de texto*

Para generar un registro del acceso a la instancia en cuestión, se hará una llamada a un documento de texto ante cada comprobación de acceso, añadiendo en el nuevo registro si se ha permitido o no la entrada y a que tarjeta. Esto se hará mediante el código que podemos ver en la figura 5.51, en el cual primero abrimos un archivo de texto especificado con el parámetro de apertura 'a', el cual nos permite abrir el documento por el final y continuar agregando información desde ahí, lo cual nos generará un registro continuo del acceso a la habitación. Después se escribe el mensaje pertinente en el archivo en función de si se ha permitido o no el acceso y se cierra el mismo.

```
archivo = open("registro.txt","a")
mensaje = "Acceso PERMITIDO al empleado con la tarjeta: "
archivo.write(mensaje + ID_empleado + '\n')
archivo.close()
```

Figura 5.51: Crear registro de acceso en un fichero de texto

5.5.2 Diagramas de flujo del programa para control de acceso

A continuación, se mostrarán los diagramas de flujo de las funciones más significativas de este software para control de acceso. estos diagramas serán los de la función main() del programa y de la función Mifare(). Primero podemos observar el diagrama de flujo del main() en la figura 5.52, en el que una vez se han declarado, definido e inicializado todos los parámetros que necesitaremos a lo largo del programa se queda en un bucle while en el que llama a la función encargada de detectar tarjetas cada 0.1 segundos. A continuación podemos ver el diagrama de flujo de la función Mifare() en la figura 5.52, que se encarga de comprobar si se ha acercado una tarjeta y actuar conforme a ello en caso de detectarla. En caso de detectar una tarjeta puede actuar según esta sea de cliente o de empleado, y dentro de cada posibilidad comprobará los permisos asignados a la tarjeta y los comprobará con los otorgados a la habitación, permitiendo o negando el paso.

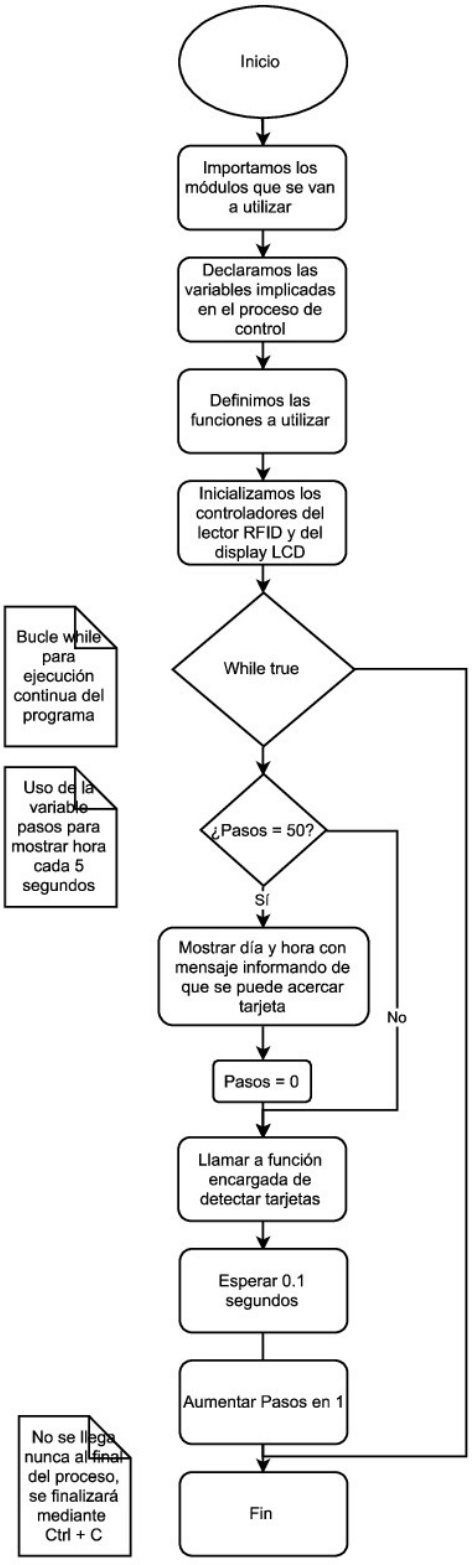


Figura 5.52: diagrama de flujo del programa en Python para control de acceso

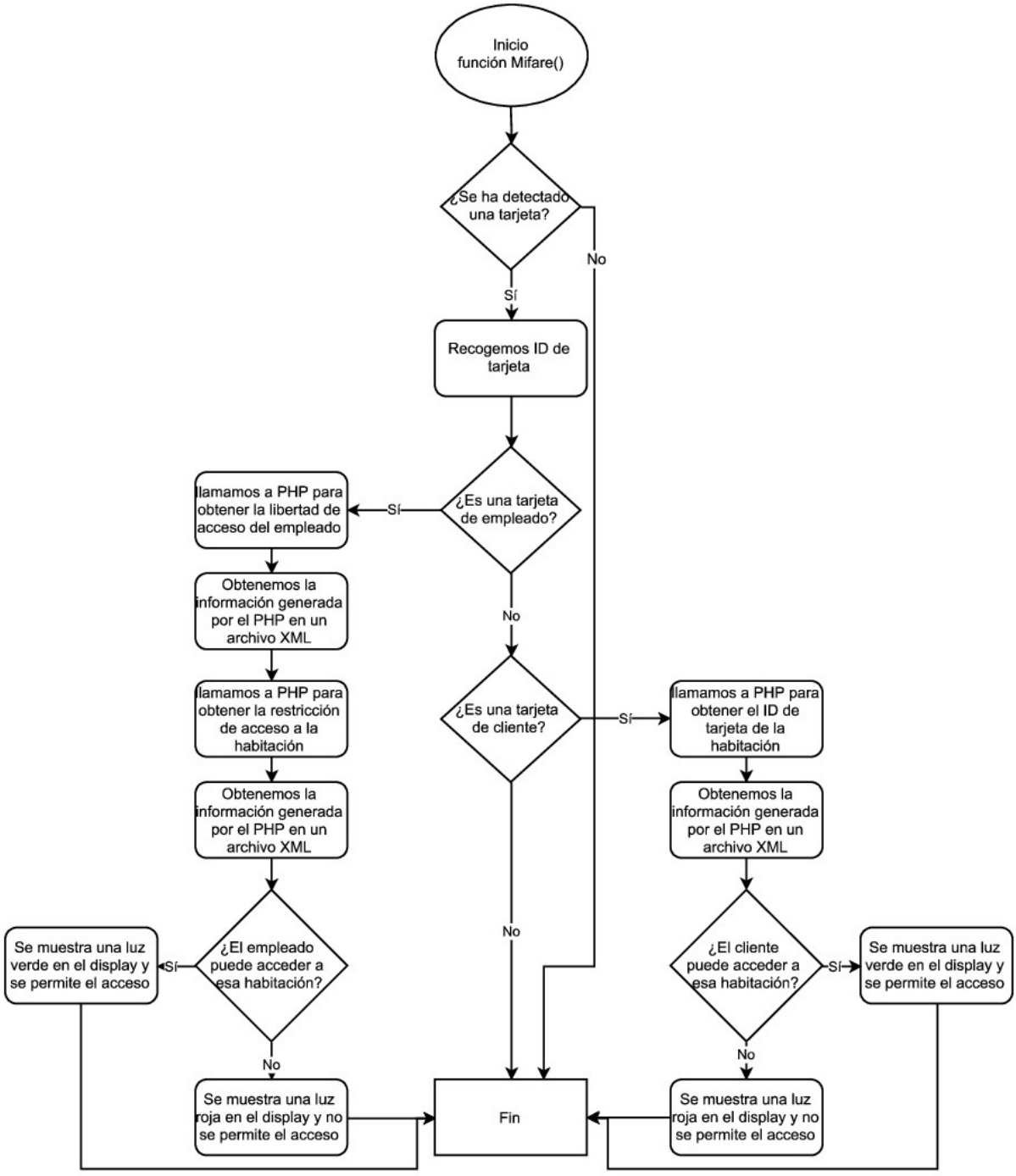


Figura 5.53: diagrama de flujo de la función Mifare()

6 Pruebas de funcionamiento del sistema

En este apartado se realizarán las pruebas necesarias desde ambos clientes del sistema y se documentarán gráficamente a fin de aportar una imagen más clara del funcionamiento del sistema en su conjunto.

6.1 Pruebas de funcionamiento de la aplicación de escritorio

Primero se realizarán las pruebas convenientes con la aplicación de escritorio y se comparará el estado de la base de datos ante cada acción, estando esta información disponible en los listados que presenta la aplicación en sus distintas ventanas. Se mostrarán las pruebas realizadas para cada uno de los botones encargados de realizar alguna función que implique una interacción con la base de datos, mostrando por orden las funciones de agregar, eliminar y modificar información en las tablas clientes, habitaciones y personal.

6.1.1 Alta de nuevo cliente

En el proceso de dar de alta a un cliente observamos en la figura 6.1 el estado de la tabla clientes en la base de datos mostrado mediante la lista que contiene la ventana Registro clientes, a la izquierda el estado inicial, y a la derecha el estado que presenta una vez hemos añadido correctamente todos los campos de la parte de "Alta de clientes" y pulsado el botón "Nuevo Cliente". En la figura 6.2 observamos como al introducir un nuevo cliente la habitación que se le asigna cambia su disponibilidad a "Ocupada", mostrando también en color rojo la habitación 201 que es en la que se ha registrado al nuevo cliente.

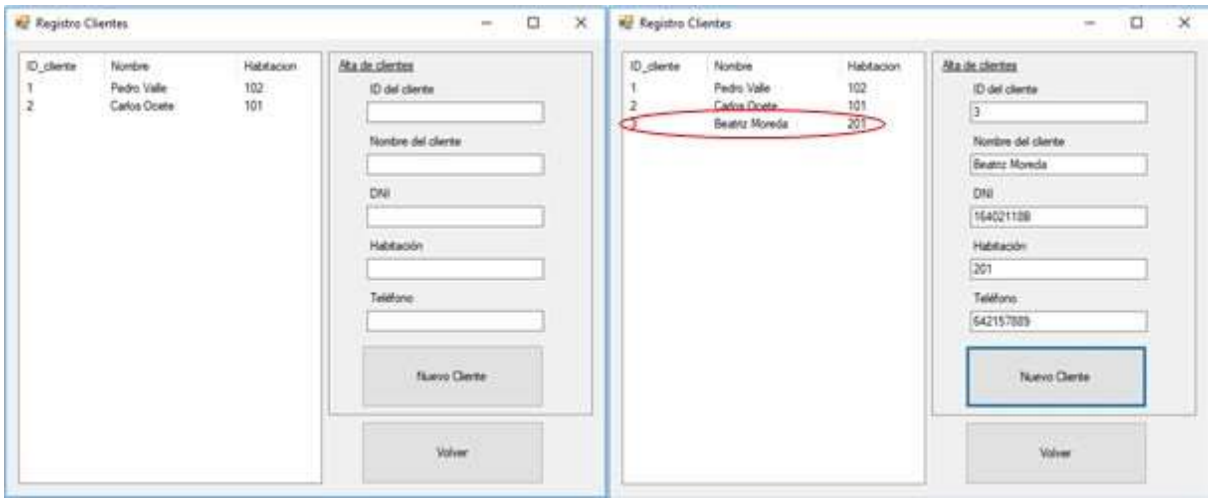


Figura 6.1: Registro de un nuevo cliente

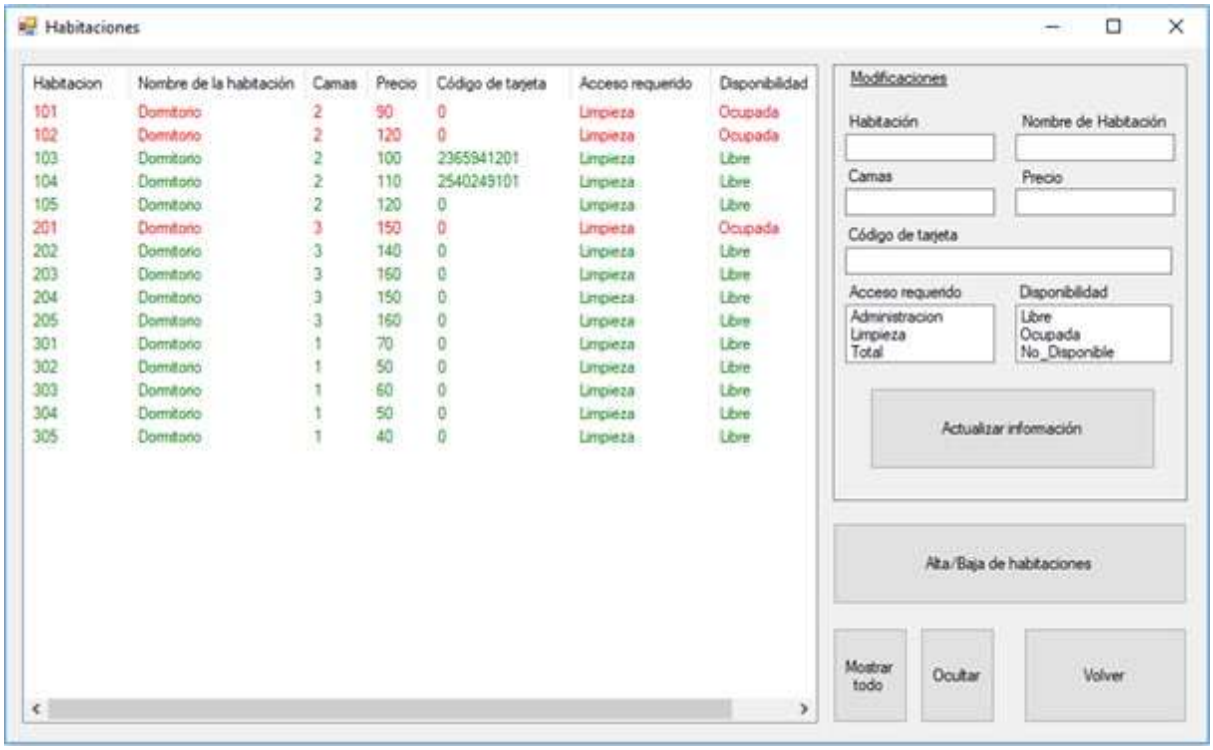


Figura 6.2: Listado de habitaciones tras registrar un nuevo cliente

6.1.2 Baja de un cliente

Dando de baja a un cliente podemos observar en la figura 6.3 cómo el cliente con ID '3' es eliminado de la base de datos cuando introducimos este ID en el cuadro correspondiente y pulsamos el botón, cambiando de forma contraria al proceso de alta de un cliente la disponibilidad de la habitación en la que se encontraba registrado a "Libre", como se puede ver en la figura 6.4.

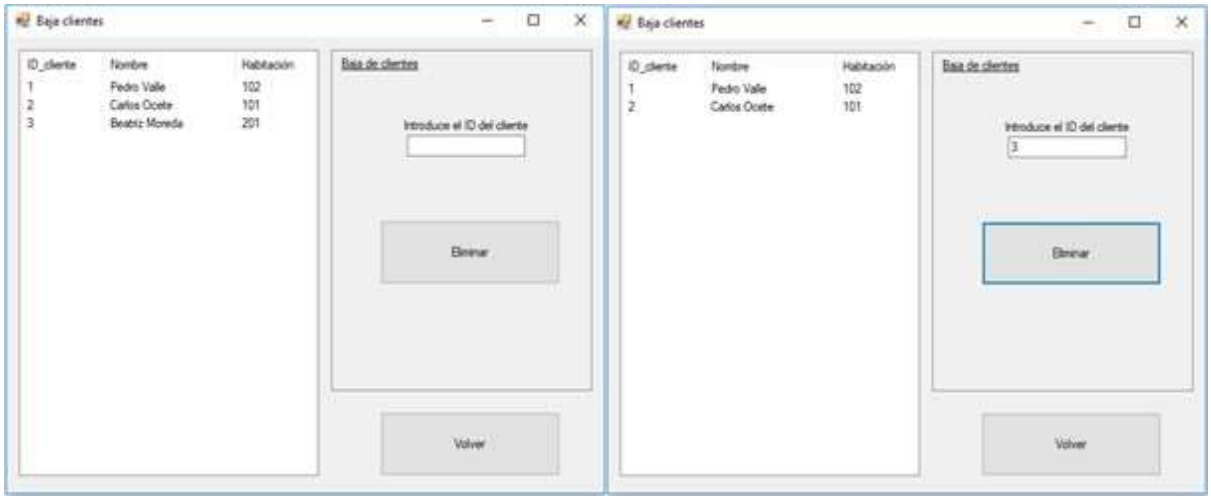


Figura 6.3: Baja de un cliente existente

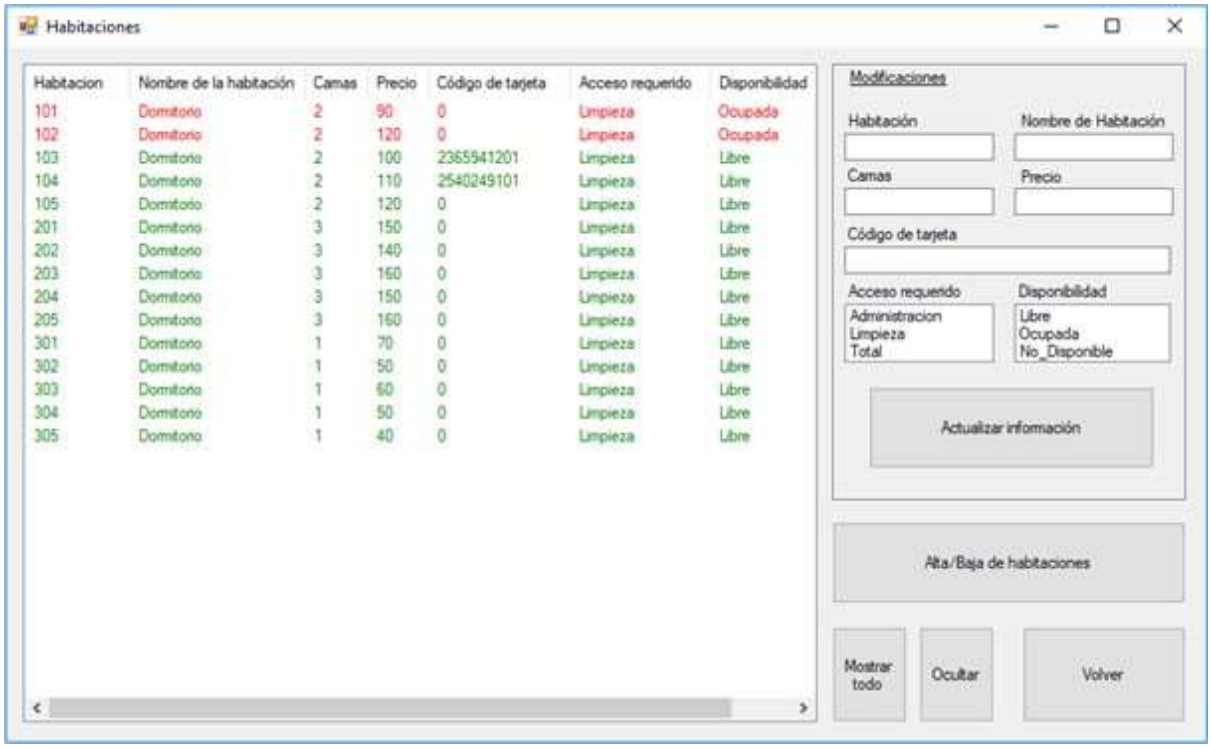


Figura 6.4: Listado de habitaciones tras dar de baja un cliente

6.1.3 Modificar información de un cliente

Para mostrar la modificación de la información de un cliente se ha decidido cambiar la habitación, para mostrar también como la disponibilidad de las habitaciones cambiar cuando se modifica este campo. En la figura 6.5 podemos observar que al cambiar la habitación del cliente 1, que se encontraba registrado en la 102 por la 105, este campo se modifica, así como la disponibilidad de estas habitaciones, como se puede observar en la figura 6.6, en la que la habitación 102 se encuentra ahora libre mientras que la 105 ha pasado a estar ocupada.

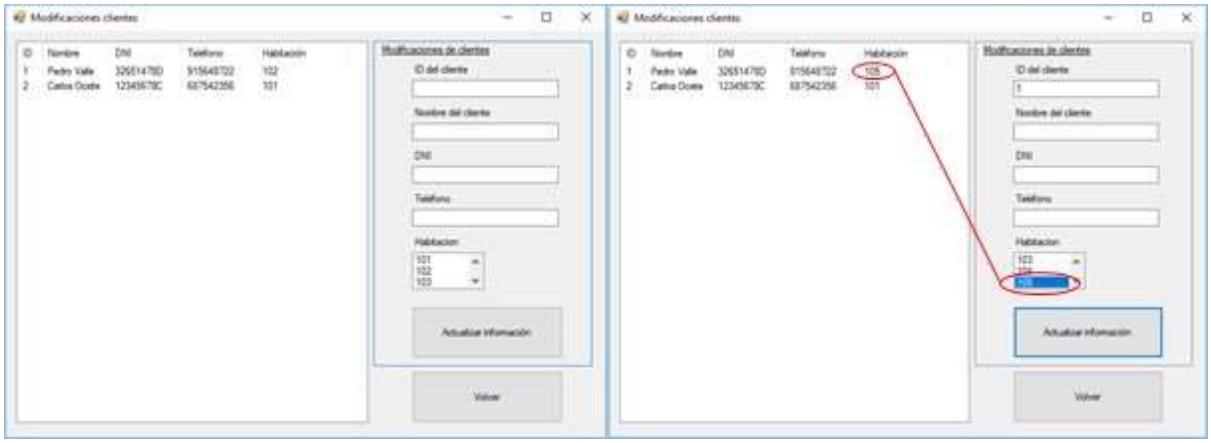


Figura 6.5: Modificación de la información de un cliente

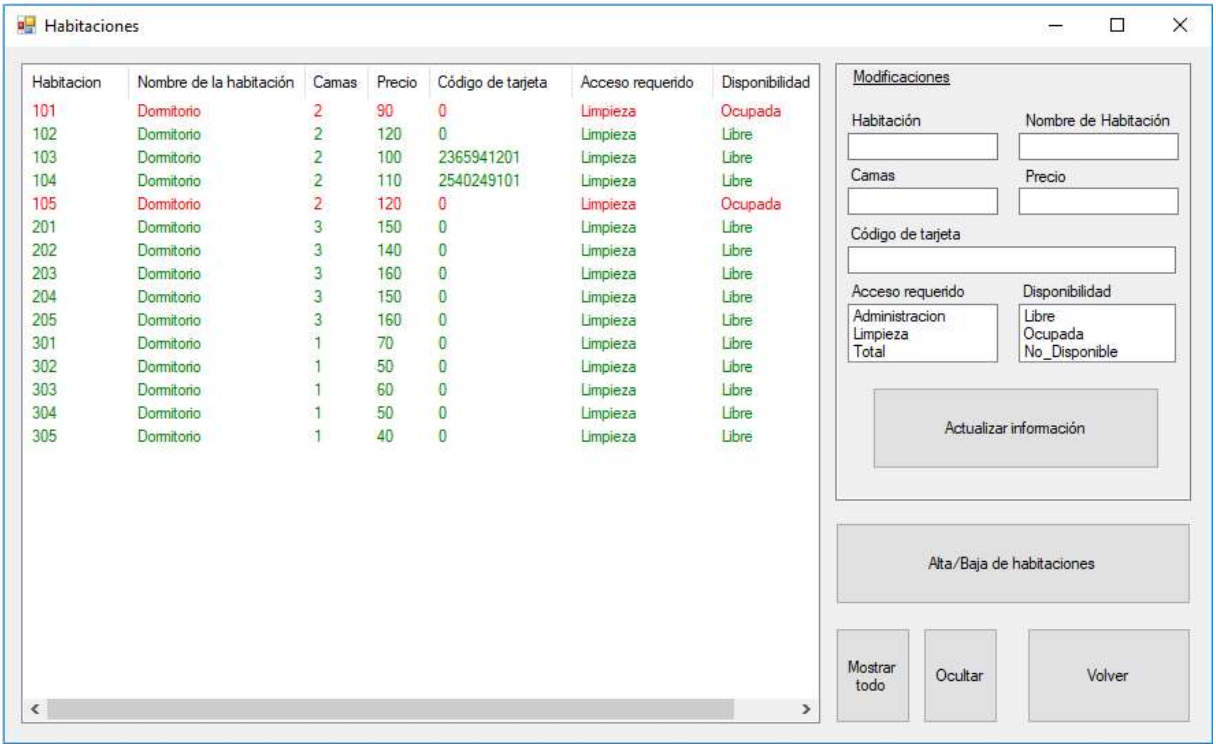


Figura 6.6: Listado de habitaciones tras modificar la habitación de un cliente

6.1.4 Alta de una nueva habitación

En el alta de habitaciones partiremos del estado anterior de habitaciones mostrado en la figura 6.6 y añadiremos una nueva habitación al hotel, una Suite con el número de habitación 401. Podemos observar en la figura 6.7 todos los campos introducidos en la parte de "Alta de habitaciones" y la nueva habitación con estos datos en sus respectivas columnas.

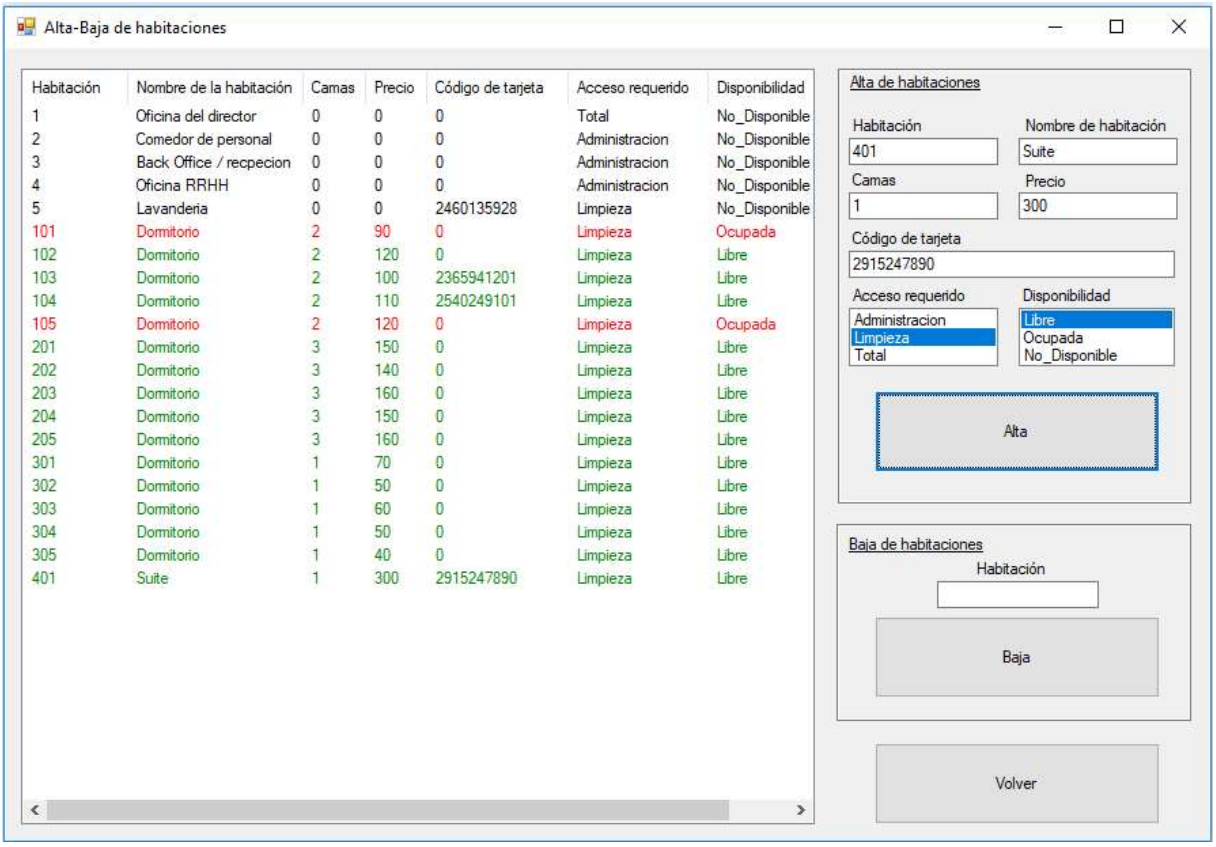


Figura 6.7: Alta de una nueva habitación

6.1.5 Baja de una habitación

En la parte de baja de habitaciones, tan solo tendremos que introducir el número de la habitación que deseamos eliminar y pulsar en el botón “Baja”, eliminándose así la habitación de la base de datos como se puede ver en la figura 6.8, en la que acabamos de dar de baja la habitación 401 que dimos de alta en el apartado anterior.

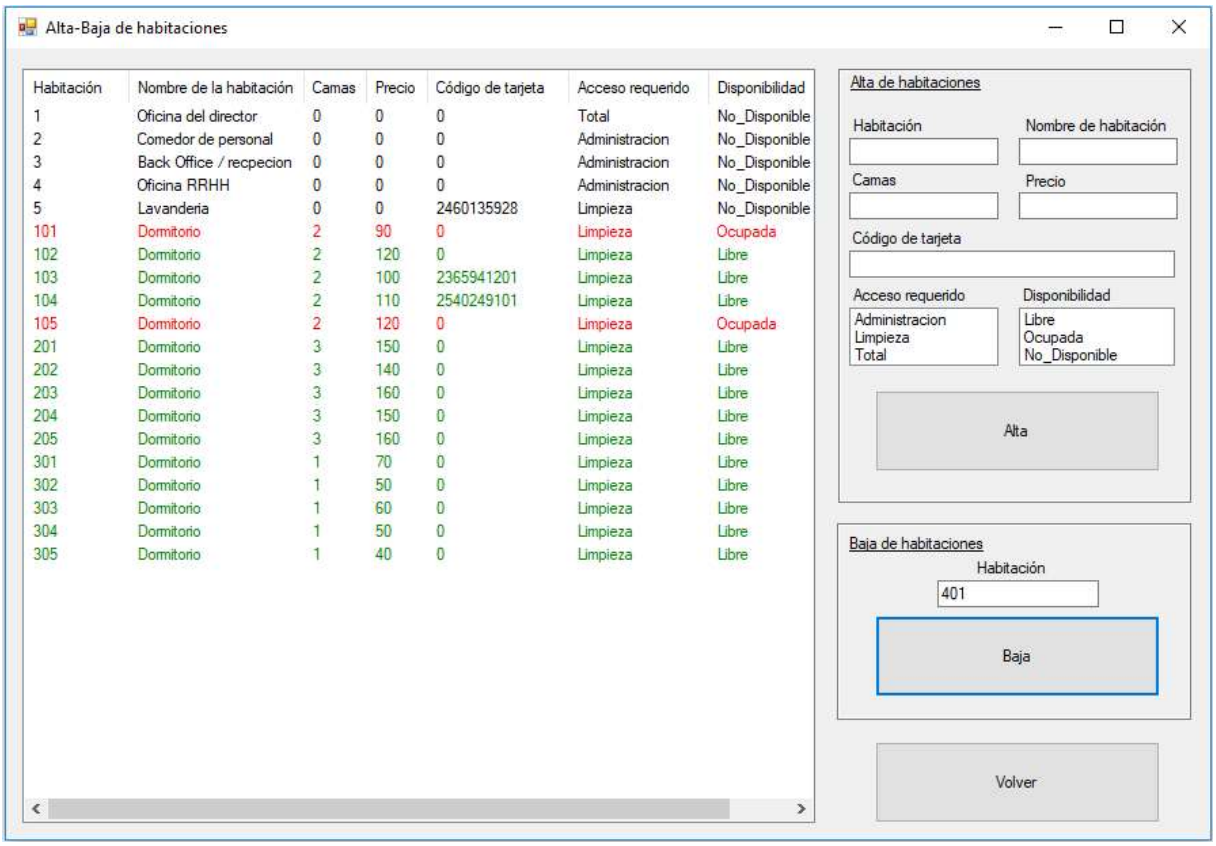


Figura 6.8: Baja de una habitación existente

6.1.6 Modificar la información de una habitación

Para modificar la información de una habitación se ha decidido modificar la disponibilidad de la misma, cambiando así también el color de toda la fila. En la figura 6.9 observamos cómo la habitación 305 que antes estaba libre pasa a estar ocupada al introducir dicho número de habitación y seleccionar su disponibilidad como "Ocupada".

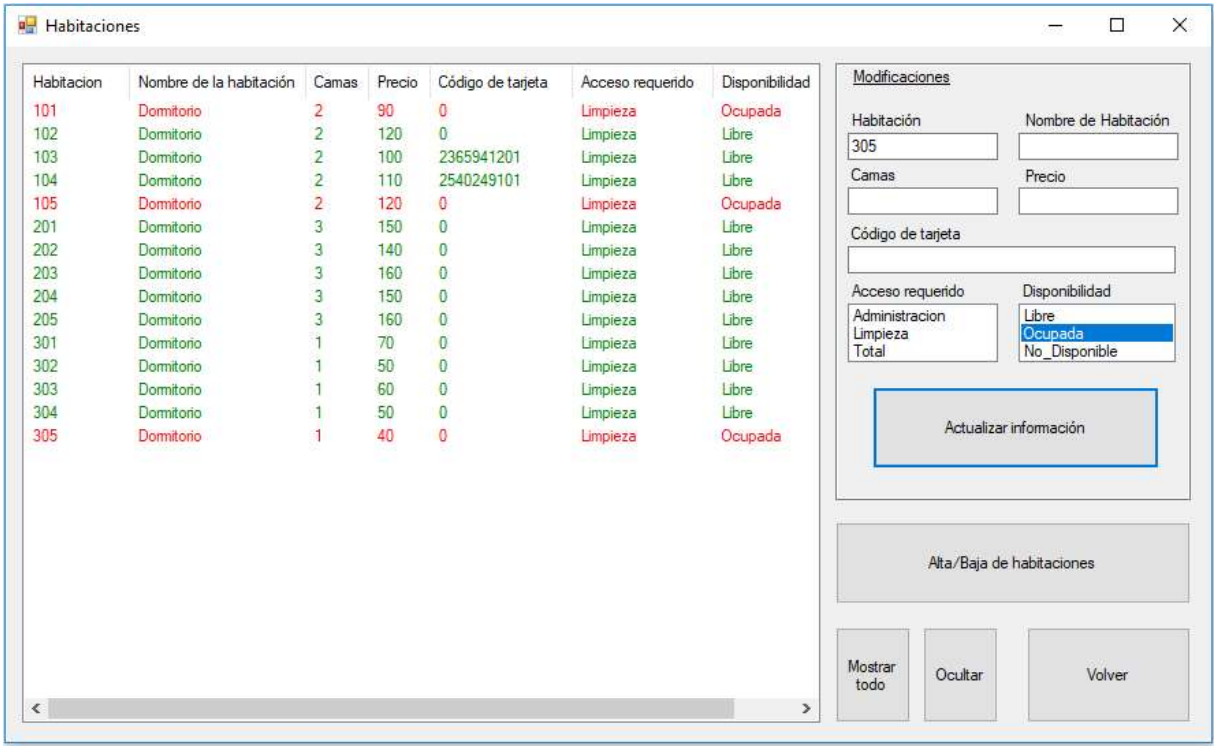


Figura 6.9: Modificar la información de una habitación

6.1.7 Alta de un nuevo empleado

En la figura 6.10 podemos ver cómo en la ventana de registro de personal podemos dar de alta un nuevo empleado de forma análoga a la mostrada en el alta de un nuevo cliente, introduciéndose este en la base de datos al introducir la información correspondiente en sus respectivos campos y pulsar el botón "Nuevo Empleado".

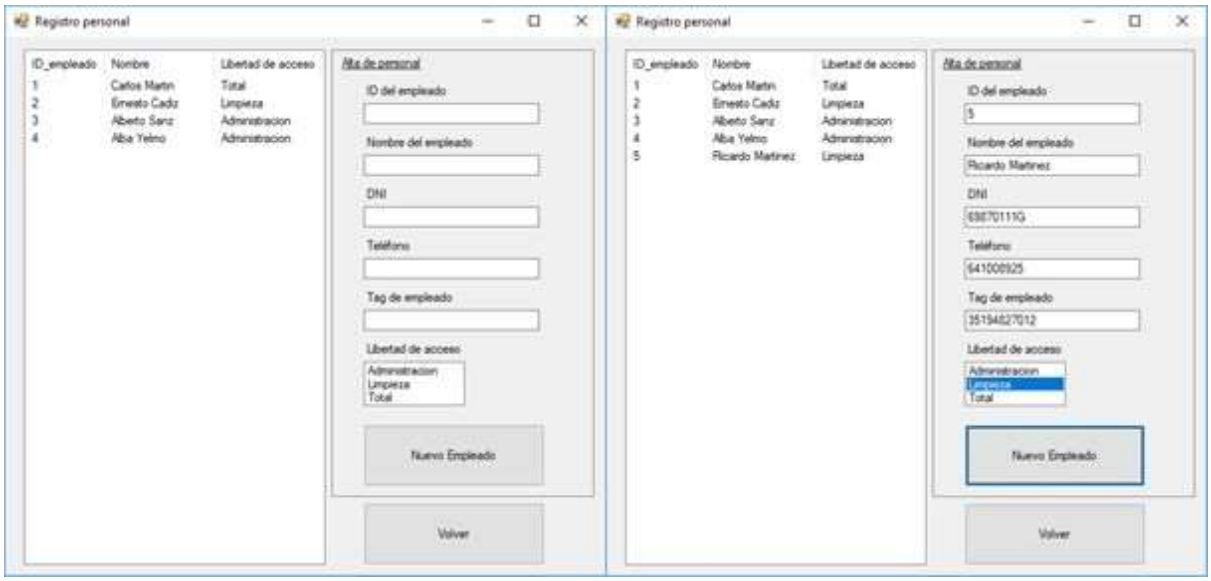


Figura 6.10: Registro de un nuevo empleado

6.1.8 Baja de un empleado

Para dar de baja un empleado, tal y como se observa en la figura 6.11, introduciremos el ID del empleado que queramos dar de baja y pulsamos en el botón "Eliminar", borrándose este de la base de datos.

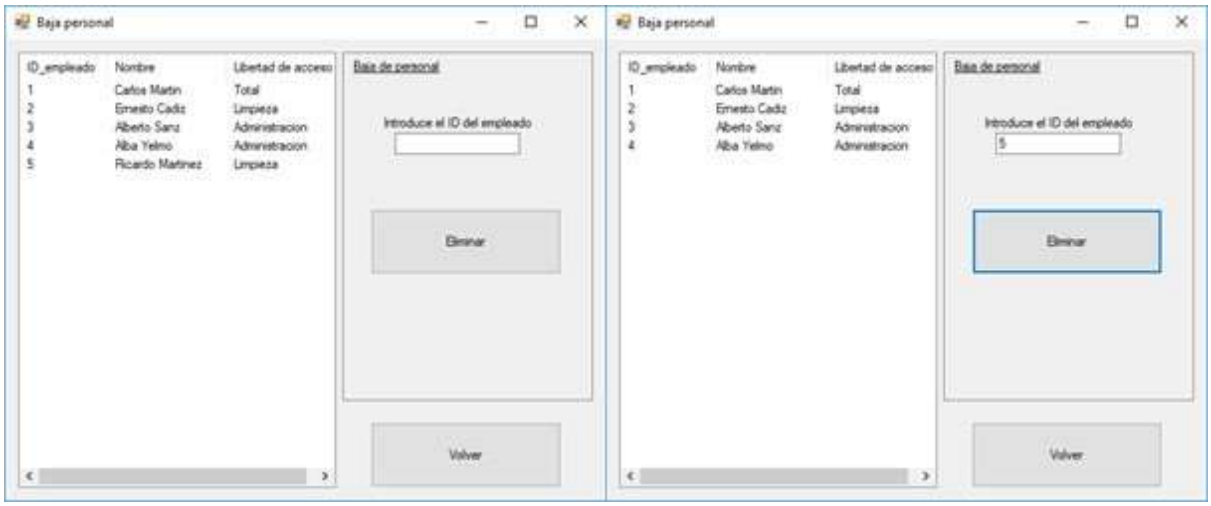


Figura 6.11: Baja de un empleado existente

6.1.9 Modificar la información de un empleado

En la modificación de la información de un empleado se ha optado por modificar el teléfono y la libertad de acceso del empleado con ID '3', viéndose en la figura 6.12 cómo tiene un teléfono móvil asignado y una libertad de acceso de Administración, y al modificar estos datos tal y como se muestra en la figura 6.13 su información cambia.

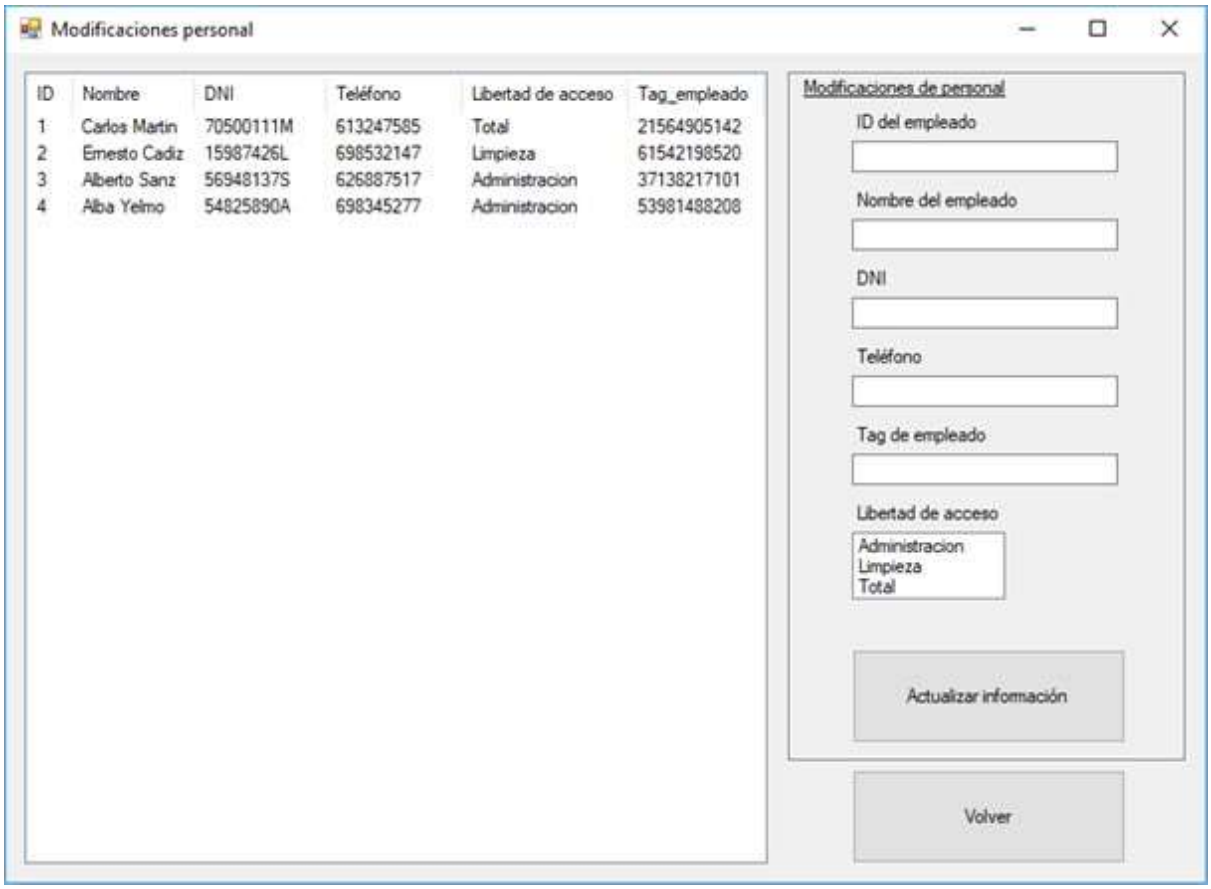


Figura 6.12: Estado antes de modificar la información de un empleado

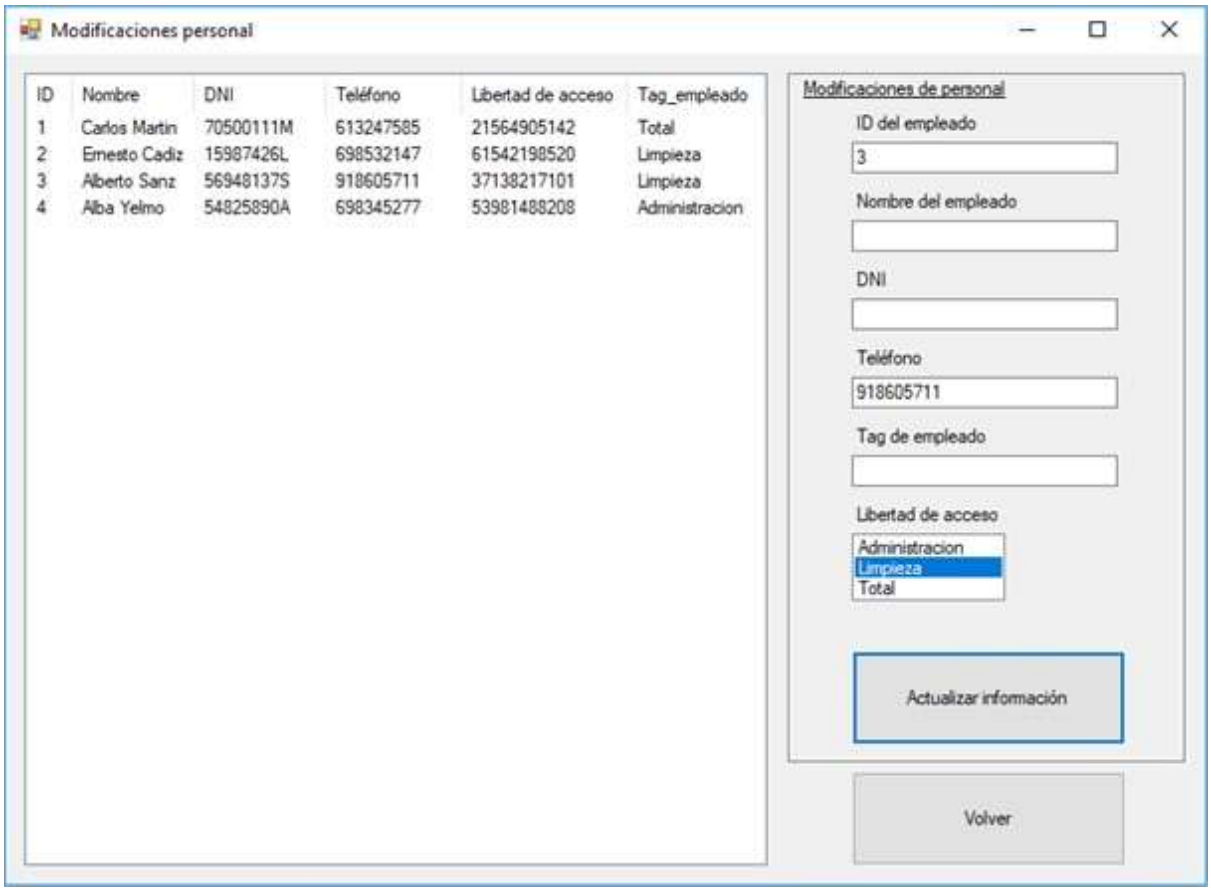


Figura 6.13: Estado después de modificar la información de un empleado

6.2 Pruebas de funcionamiento del sistema de control de acceso

Las pruebas de funcionamiento del sistema de control de acceso consistirán en imágenes de una serie de supuestos a los que el sistema de control de acceso responderá permitiendo el acceso y mostrando una luz verde en el display LCD o negándolo y mostrando una luz roja en su lugar. Para estas pruebas se utilizará la tarjeta blanca a modo de tarjeta de cliente con el código que abre la habitación 104, mientras que el tag azul será una tarjeta de usuario, asignada al empleado Alberto Sanz, con una libertad de acceso de tipo “Administración”, por lo que podrá acceder a las instancias destinadas a uso del personal, pero no a los dormitorios de los clientes, que requieren una libertad de acceso de tipo “Limpieza”. Ambos identificadores se pueden ver en la figura 6.14.

Figura 6.14: Tarjeta de cliente y tag de empleado

6.2.1 Acceso a dormitorio no asignado

Primero se configura el programa de control de acceso para que actúe como la puerta del dormitorio 103, cuyo código no está asignado a la tarjeta cliente y al cual el empleado no podrá acceder con su tag cuya libertad de acceso es "Administración". En la figura 6.15 y 6.16 respectivamente podemos ver cómo, en efecto, ni la tarjeta blanca ni el tag de empleado pueden acceder a la habitación.

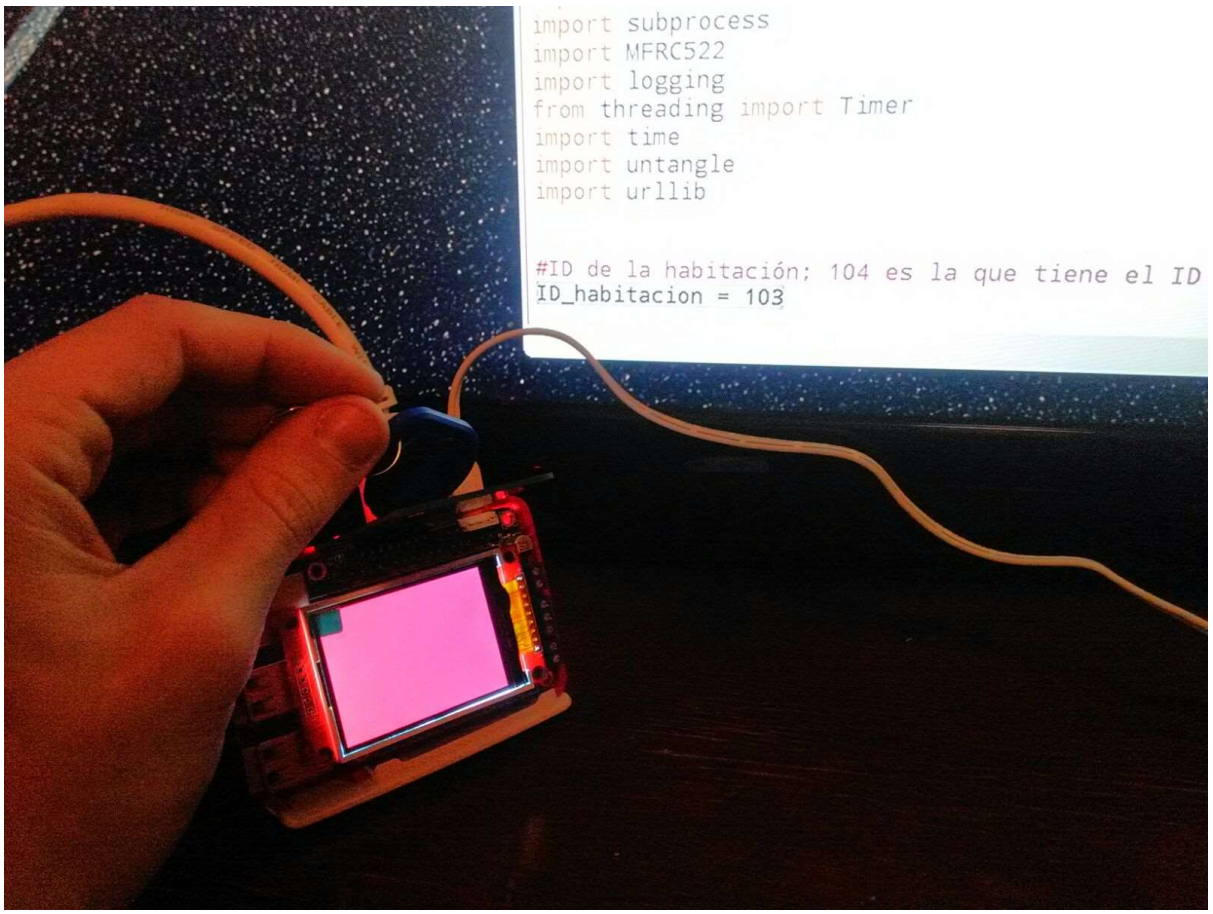


Figura 6.15: Intento de acceso con Tag de empleado a dormitorio 103

Figura 6.16: Intento de acceso con tarjeta de cliente a dormitorio 103

6.2.2 Acceso a dormitorio asignado

En este segundo caso se configura el programa de control de acceso para actuar como la puerta del dormitorio 104, cuyo código de acceso está asignado a la tarjeta blanca y por tanto se le permite el acceso mientras que el tag de empleado sigue sin poder acceder al dormitorio. Podemos comprobar en las figuras 6.17 y 6.18 que el sistema responde correctamente a estos supuestos.

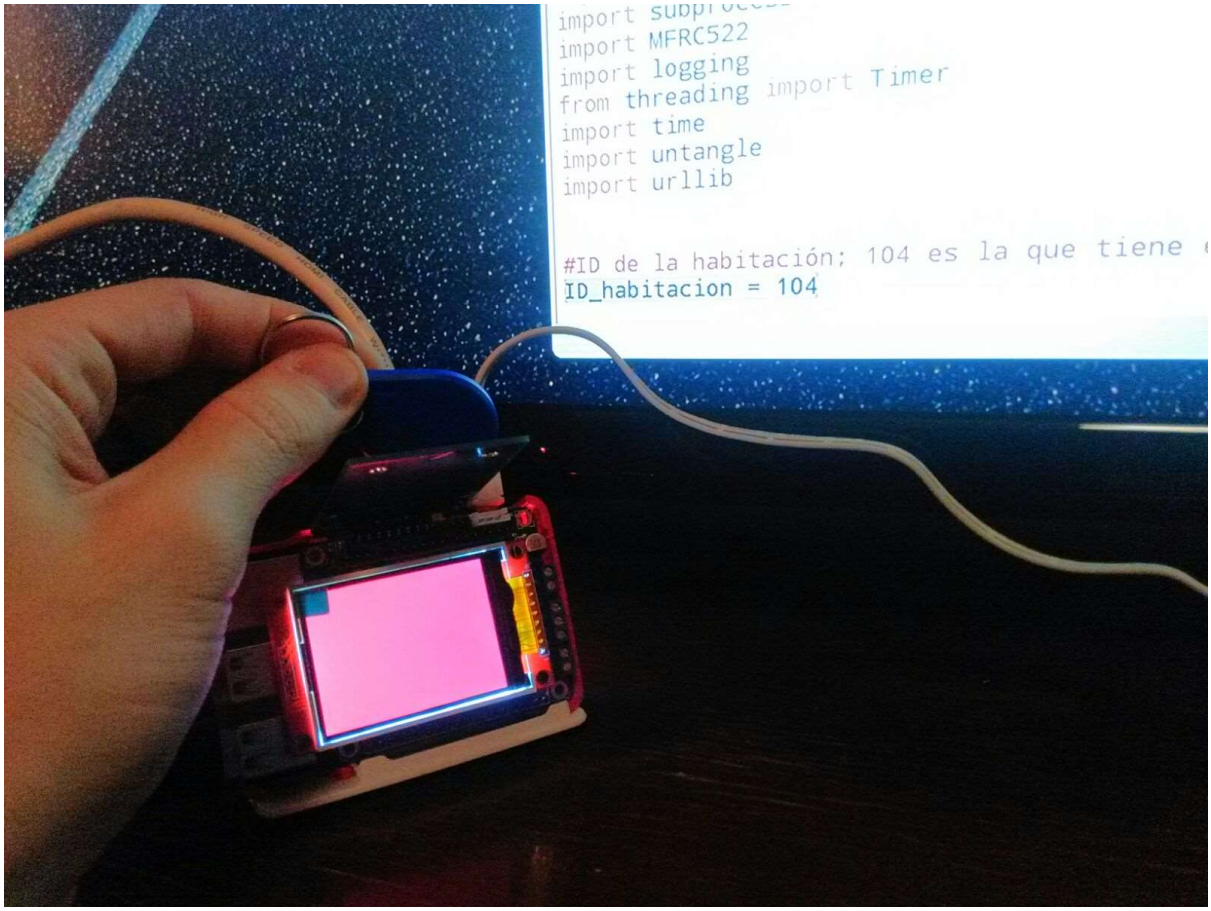


Figura 6.17: Intento de acceso con Tag de empleado a dormitorio 104

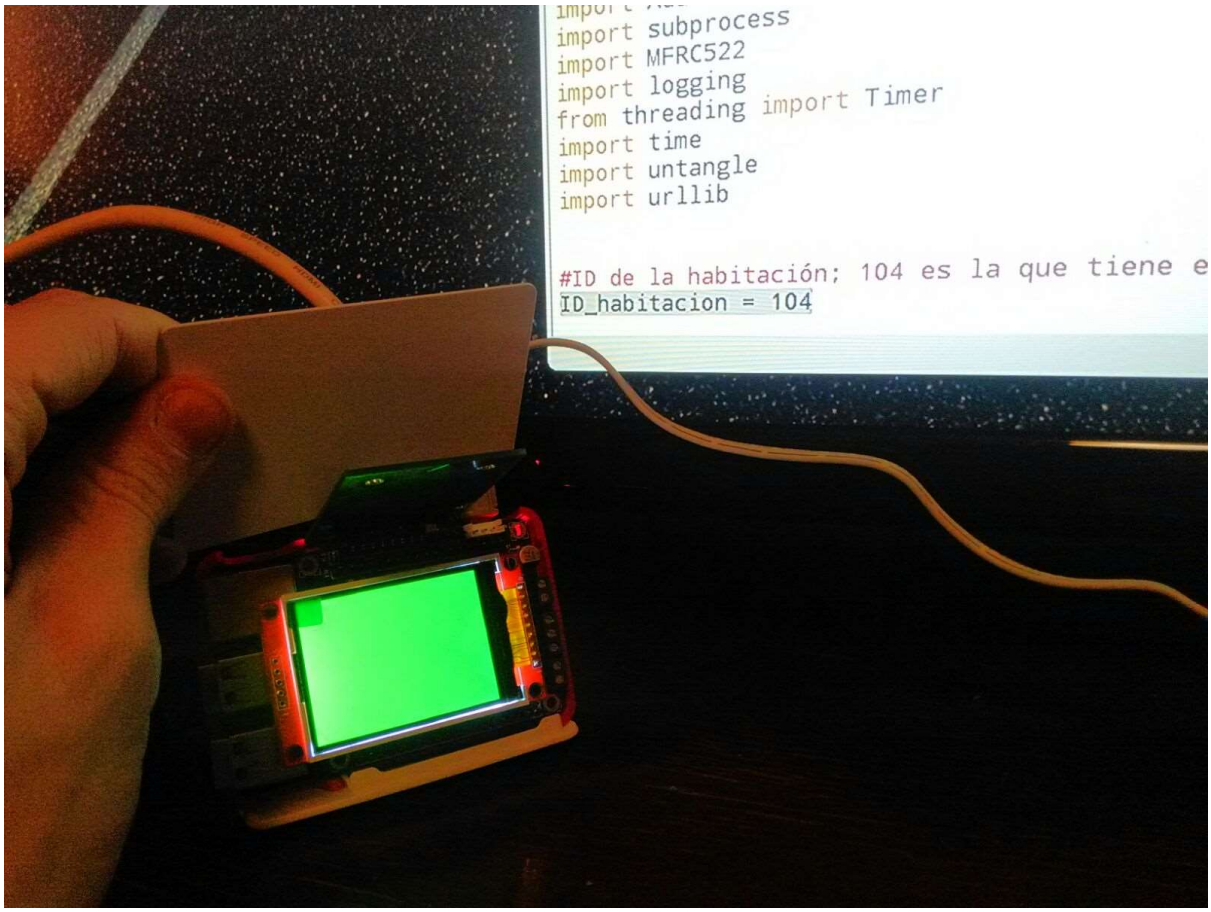


Figura 6.18: Intento de acceso con tarjeta de cliente a dormitorio 104

6.2.3 Acceso a instancia de uso del personal

A continuación, se configura el programa de control de acceso para que actúa como la puerta de la habitación 4, que está asignada a la oficina de recursos humanos y, por tanto, la tarjeta de cliente no podrá acceder a esta instancia mientras que el tag de empleado si que tendrá acceso a la habitación. En las figuras 6.19 y 6.20 comprobamos que, en efecto, el empleado puede acceder a la habitación mientras que se niega el acceso al cliente.

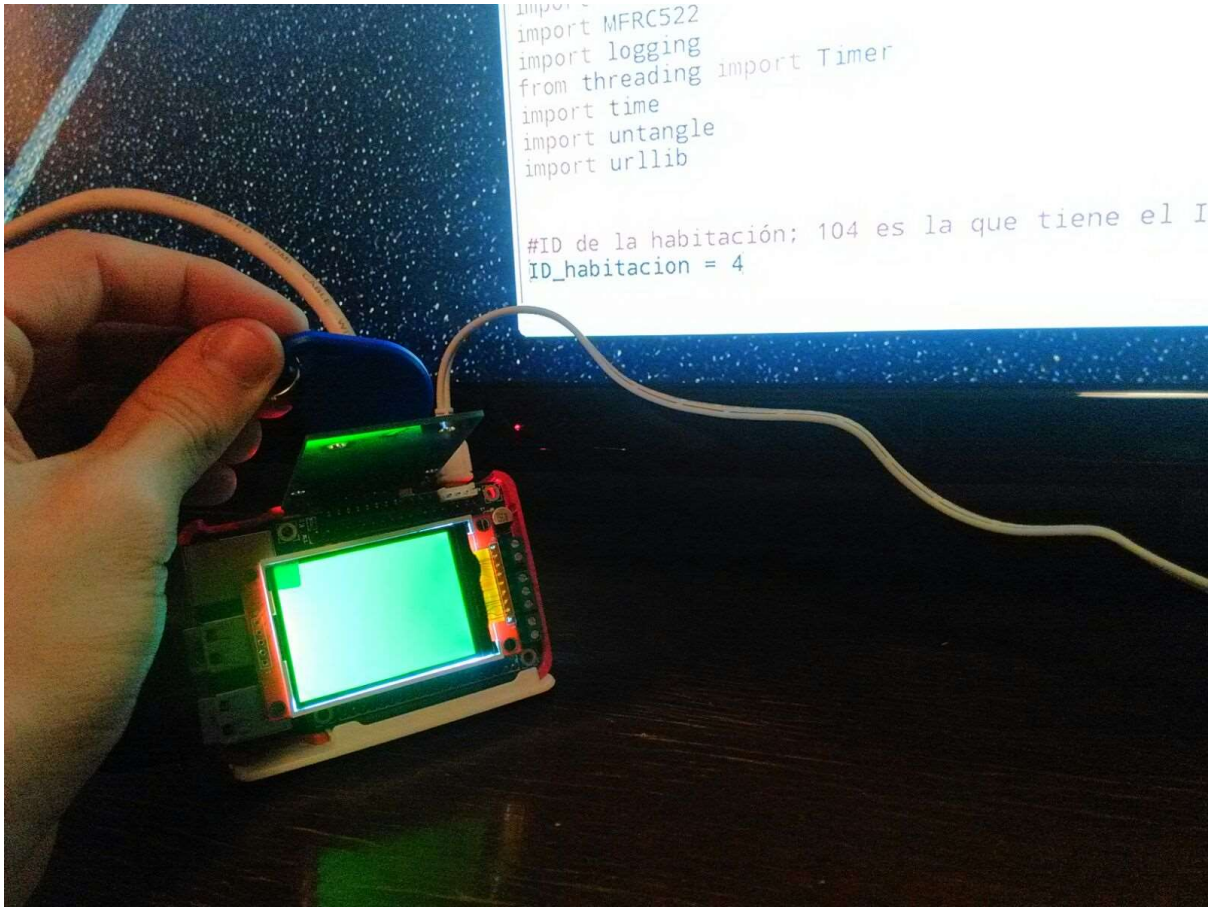


Figura 6.19: Intento de acceso con Tag de empleado a habitación 4, despacho de RRHH

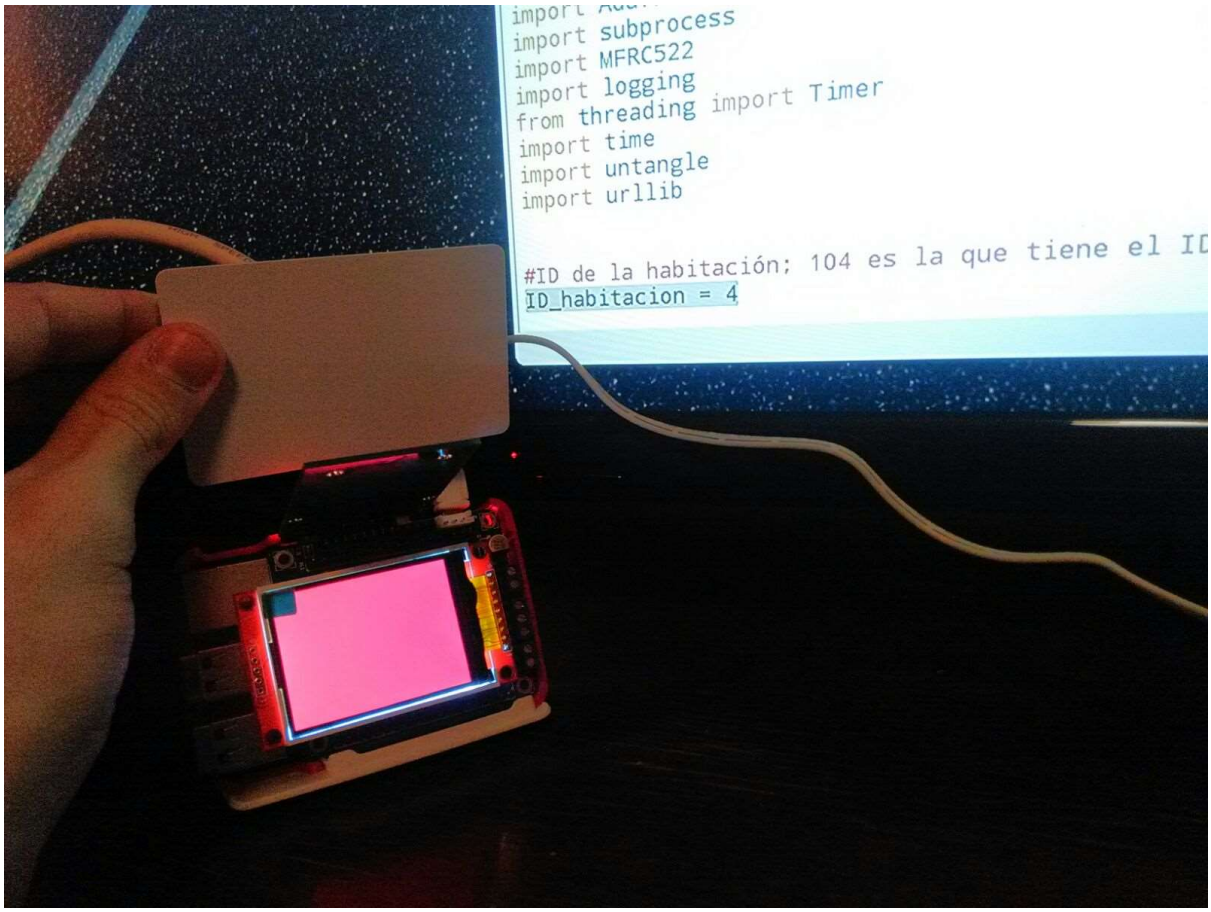


Figura 6.20: Intento de acceso con tarjeta de cliente a habitación 4, despacho de RRHH

6.2.4 Acceso a dormitorio asignado con cambio desde la BD

Por último, volveremos al segundo caso, en el que se configura el programa de control de acceso para que actúa como la puerta del dormitorio 104, de modo que el cliente podrá acceder a la misma mientras que el empleado no, pero en este caso vamos a modificar también la libertad de acceso de nuestro empleado modificando el campo directamente desde la base de datos como se puede ver en la figura 6.21, observando que una vez realizado este cambio se permitirá el acceso a ambos identificadores, como podemos ver en las figuras 6.22 y 6.23.

ID_Empleado	Nombre	DNI	Telefono	Libertad_acceso	Tag_empleado
1	Carlos Martin	70500111M	613247585	Total	21564905142
2	Ernesto Cadiz	15987426L	698532147	Limpieza	61542198520
3	Alberto Sanz	56948137S	918605711	Administracion	37138217101
4	Alba Yelmo	54825890A	698345277	Administracion	53981488208

ID_Empleado	Nombre	DNI	Telefono	Libertad_acceso	Tag_empleado
1	Carlos Martin	70500111M	613247585	Total	21564905142
2	Ernesto Cadiz	15987426L	698532147	Limpieza	61542198520
3	Alberto Sanz	56948137S	918605711	Limpieza	37138217101
4	Alba Yelmo	54825890A	698345277	Administracion	53981488208

Figura 6.21: Cambio de libertad de acceso de Administración a Limpieza del empleado asignado al Tag utilizado durante las pruebas

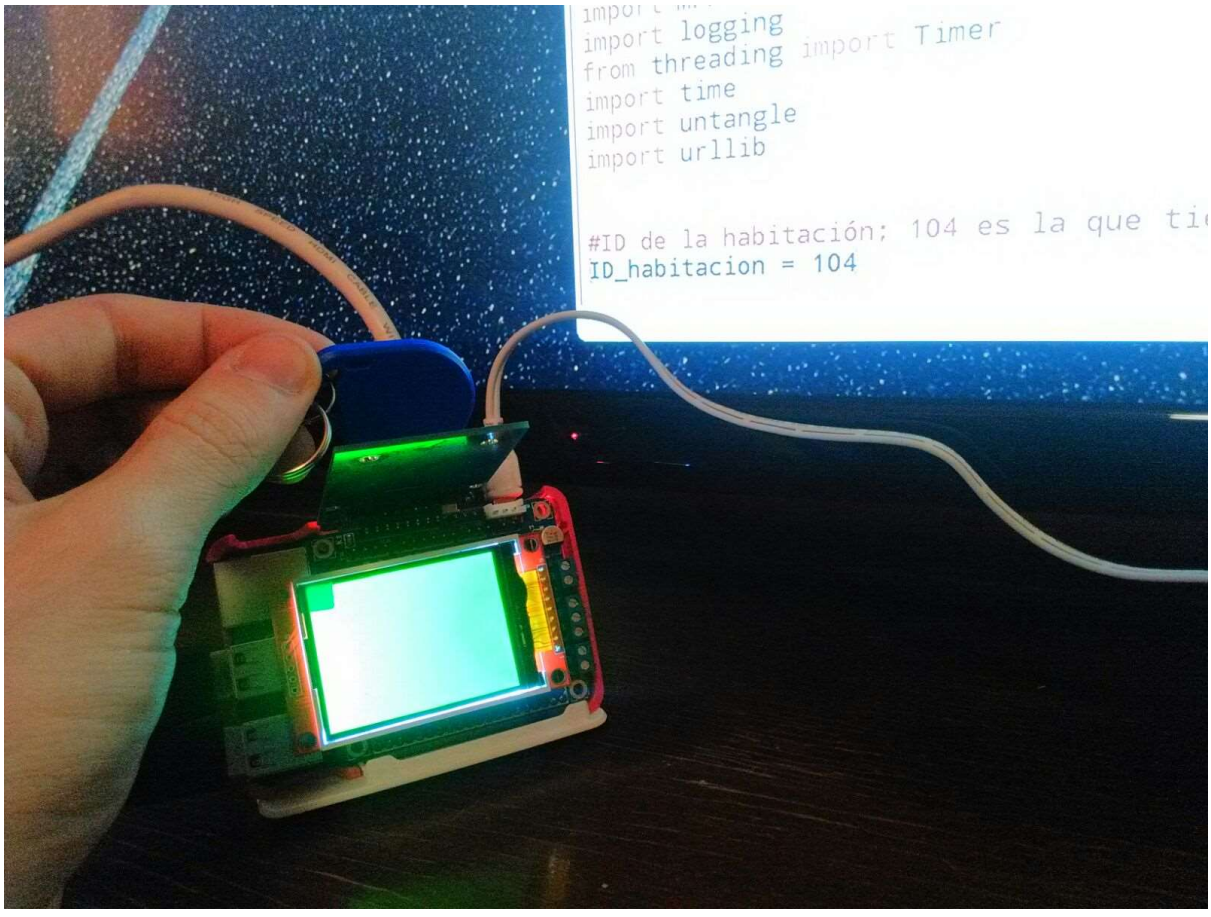


Figura 6.22: Intento de acceso con Tag de empleado a dormitorio 104

Figura 6.23: Intento de acceso con tarjeta de cliente a dormitorio 104

7 Conclusiones

En este capítulo se expondrán las conclusiones extraídas tras la finalización del proyecto, tanto a nivel personal como desde el punto de vista del cumplimiento de los objetivos marcados al inicio del proyecto y el establecimiento de nuevas líneas de investigación a partir de la realización del sistema expuesto en este documento.

La motivación de este documento fue, entre o tras cosas, el interés por la tecnología Raspberry Pi y el software destinado a las interfaces web. Tras finalizar el proyecto podemos concluir que se han adquirido estos conocimientos, así como un aumento en el interés por estos campos y algunos más específicos vistos durante el desarrollo del sistema, como el lenguaje Python. También podemos concluir que se han adquirido una gran cantidad de conocimientos no mostrados durante el grado, sin embargo, los conocimientos adquiridos en otros campos han significado una clara ayuda a la hora de aprender, entender y aplicar estos nuevos conocimientos.

7.1 Objetivos alcanzados

A lo largo de este documento se puede comprobar como los objetivos establecidos en un principio se han cumplido obteniendo resultados satisfactorios. Se ha conseguido desarrollar un sistema que gestione toda la información necesaria sobre empleados, clientes y habitaciones de un hotel, a la vez que utilice esta información para controlar el acceso a las instancias del hotel. Se ha cumplido, por tanto, el objetivo principal del proyecto, pero también todos los objetivos segundarios establecidos al inicio que se mencionan a continuación:

- ✓ Utilizar un servidor web para alojar toda la información almacenada
- ✓ Crear una aplicación de gestión de la información con una interfaz intuitiva
- ✓ Agregar un método de comunicación del usuario con el sistema de control de acceso

Se ha conseguido alojar toda la información del sistema en un servidor web, se ha generado una aplicación de escritorio funcional e intuitiva que permite gestionar toda la información del hotel y se ha agregado y configurado un display LCD que permite al usuario saber si puede acceder o no a una instancia cuando aproxima su tarjeta de identificación.

También se han llevado a cabo otros objetivos surgidos y superados durante el desarrollo del proceso, como realizar la conexión con la base de datos mediante scripts PHP que reciben los parámetros necesarios mediante POST, agregando así un extra de seguridad en las comunicaciones del sistema. También se ha añadido un registro al sistema de control de acceso, que guarda la información de quien entra o intenta entrar a una habitación con su respectivo ID de tarjeta.

7.2 Trabajos futuros

El sistema desarrollado cumple con todos los objetivos marcados inicialmente, sin embargo, siempre es posible ampliar y mejorar un sistema electrónico. En este punto hablaremos de las posibles mejoras y/o ampliaciones que se plantean para este proyecto en particular a fin de mejorar su funcionalidad o capacidades en algún aspecto.

Por una parte, en lo referente a la aplicación de escritorio, se podría desarrollar un cuarto botón en la pantalla de inicio que mostrase el estado de todas las cerraduras del sistema, indicando si está activa o no y cuando fue el último acceso a la instancia de controla.

En cuanto a la base de datos, una mejora que hacer a esta parte sería implementar un sistema de recogida de fechas y horas, para poder controlar mejor el flujo de clientes en el hotel y facilitar otras ampliaciones que se comentarán a continuación.

Por último, se propone un control de acceso más exhaustivo en cuanto a las horas de acceso a cada instancia, con un correcto registro de fechas se podría implementar una ampliación de las distintas libertades de acceso, permitiendo, por ejemplo, el acceso a aquellos con libertad de limpieza acceder a los dormitorios tan sólo en cierta franja horaria o limitando el acceso a las habitaciones de uso administrativo al horario laboral.

Bibliografía

[1] Maestros del web. ¿Qué son las bases de datos? http://www.maestrosdelweb.com/que-son-las-bases-de-datos/ Consultado: agosto 2017.

[2] Wikipedia. SQL. https://es.wikipedia.org/wiki/SQL. Consultado: julio 2017.

[3] El blog del Informatico. ¿Qué es XAMPP? http://blogdelinformatico-reizer.blogspot.com.es/2015/11/que-es-xampp.html. Consultado: julio 2017.

[4] Manual de PHP. https://secure.php.net/manual/es/getting-started.php. Consultado: julio 2017.

[5] Microsoft. Introducción a Visual Studio. https://msdn.microsoft.com/es-es/library/fx6bk1f4(v=vs.100).aspx. Consultado: julio 2017.

[6] Wikipedia. C Sharp. https://es.wikipedia.org/wiki/C_Sharp. Consultado: julio 2017.

[7] Microsoft Developer Network. Información general sobre formularios Windows Forms. https://msdn.microsoft.com/es-es/library/8bxxy49h(v=vs.110).aspx. Consultado: julio 2017.

[8] Wikipedia. Raspbian. https://es.wikipedia.org/wiki/Raspbian Consultado: gosto 2017

[9] El modelo cliente-servidor. http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/cliente-servidor.html. Consultado: agosto 2017.

[10] Microsoft. Visual Basic. https://docs.microsoft.com/es-es/dotnet/visual-basic/. Consultado: agosto 2017.

[11] Wikipedia. Visual C++. https://es.wikipedia.org/wiki/Visual_C%2B%2B. Consultado: agosto 2017.

[12] Microsoft Developer Network. Visual C#. https://msdn.microsoft.com/es-es/library/aa287558(v=vs.71).aspx. Consultado: agosto 2017.

[13] Arduino.cl ¿Qué es Arduino? http://arduino.cl/que-es-arduino/ Consultado: agosto 2017

[14] MyNewArduino. Beagle Bone. https://minewarduino.wordpress.com/.

[15] Instalación y uso de XAMPP en Windows. http://www.mclibre.org/consultar/php/otros/xampp_instalacion_windows.html. Consultado: agosto 2017

[16] https://upload.wikimedia.org/wikipedia/commons/1/1c/Cliente-Servidor.png. Consultado: agosto 2017

[17] https://www.adafruit.com/product/3055. Consultado: “agosto 2017”

[18] IngeniApp.com. http://www.ingeniapp.com. Consultado: “agosto 2017”

Anexo A: Planificación y Presupuesto

A continuación se va a llevar a cabo un desglose de las tareas que se han realizado a lo largo de este trabajo fin de grado, lo que facilitará posteriormente un cálculo aproximado sobre su coste.

A.1 Planificación

Debido a la complejidad de un trabajo de estas características se ha optado por dividirlo en distintas fases, las cuales se van a comentar a continuación:

Fase 1: Documentación inicial

I. Estudio de HTML y PHP, SQL, C#, manejo de Raspbian y Python (45 horas)

II. Preparación de herramientas de trabajo (5 horas)

III. Búsqueda de tutoriales y realización de aplicaciones simples (25 horas)

Fase 2: Desarrollo del sistema

I. Desarrollo de la base de datos (10 horas)

II. Desarrollo de aplicación de escritorio y scripts PHP para la comunicación (70 horas)

III. Configuración de Raspberry y desarrollo de software de control de acceso (45 horas)

Fase 3: Pruebas en un dispositivo real

I. Pruebas de aplicación de escritorio y su conexión con DB (10 horas)

II. Pruebas de aplicación de Python y su conexión con DB (10 horas)

III. Depuración de software y corrección de errores (20 horas)

Fase 4: Elaboración de la memoria

I. Redacción de la memoria (50 horas)

II. Corrección y maquetación (10 horas)

Tabla 1 - Desglose de tareas

FASES	HORAS EMPLEADAS
Documentación inicial	75
Desarrollo de la aplicación	125
Pruebas en un dispositivo real	40
Elaboración de la memoria	60
TOTAL	**300**

A.2 Presupuesto del Trabajo Fin de Grado

A.2.1 Costes materiales

El material necesario para el desarrollo del sistema ha sido un ordenador para generar el software necesario y alojar el servidor interno en el cual (prestaciones), una Raspberry pi, una pantalla con HDMI para visualizar la Raspberry, un lector RFID de tarjetas sin contacto con sus respectivas tarjetas y un display LCD de 2.2", así como una licencia de Visual Studio Professional 2017. Considerando un periodo de amortización de 3 años en los dispositivos y el tiempo que se han empleado en el desarrollo del sistema, unos 6 meses, se pueden ver los costes materiales en la tabla 2.

Tabla 2 – Costes Materiales

CONCEPTO	PRECIO (€)
Ordenador de altas prestaciones	116.66
Raspberry Pi 3 starter kit	125.10
Kit desarrollo IRP102	64.90
Licencia Visual Studio 2017	474.40
TOTAL	**781.06**

A.2.2 Costes de personal

Para la realización de este trabajo, ha sido necesaria la presencia de un jefe de proyecto y un ingeniero. Los costes derivados de este trabajo se pueden ver en la tabla 3

Tabla 3 – Costes de Personal

OCUPACIÓN	HORAS	PRECIO/HORA	IMPORTE (€)
Jefe de proyecto	25	90	2250
Ingeniero	275	60	16500
TOTAL	**300**		**18750**

A.2.3 Costes totales

Los costes totales del proyecto se muestran en la tabla 4.

Tabla 4 – Costes Totales

CONCEPTO	PRECIO (€)
Costes de materiales	781.06
Costes de personal	18750.00
Costes indirectos (20%)	3750.00
Subtotal	23281.06
IVA (18%)	4190.60
TOTAL	**27471.66**

El coste total del proyecto es de VEINTISSIETE MIL CUATRO CUATROCIENTOS SETENTA Y UN EUROS CON SESENTA Y SEIS CÉNTIMOS.

Leganés, 25 de septiembre de 2017

Rubén Antonio Martínez Domínguez